THESIS

IMPLEMENTING REMOTE IMAGE CAPTURE/CONTROL IN A WIRELESS SENSOR NETWORK UTILIZING THE IEEE 802.15.4 STANDARD

by

Daniel E. Krehling

September 2009

Thesis Co-Advisors:	John Gibson
	Gurminder Singh

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2009	**3. REPORT TYPE AND DATES COVERED** Master's Thesis
4. TITLE AND SUBTITLE Implementing Remote Image Capture/Control in a Wireless Sensor Network Utilizing the IEEE 802.15.4 Standard		**5. FUNDING NUMBERS**
6. AUTHOR(S) Daniel E. Krehling		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Today's warfighter requires an in-depth view of the battle space in order to best plan for future operations, assess the current operating environment, and prevent or respond to attacks. The deployment and use of wireless sensor devices could serve as a force multiplier by enhancing a commander's security posture, providing a view of t he current environment, and gathering intelligence for analysis. The use of low- power imaging devices, coupled with the flexibility provided by a wire less sensor network, could provide such enhancements.

The objective of th is research was to explore the feasibility of remote management and control of a low-power/low- cost wireless sensor network by implementing a point-to-point wireless network utilizing IEEE 802.15.4 equipped devices to control, capture, and transfer image data from a remote sensor node to the controlling host. This platform was used to test the viability of the system at various ranges and operating environments. The results demonstrated that the IEEE 802.15.4 compliant devices used in this research are able to operate over long distances (1000 meters) , in harsh RF environments, with a high degree of reliability.

14. SUBJECT TERMS IEEE 802.15.4, Wireless Sensor Network, Remote Imaging, Wireless			**15. NUMBER OF PAGES** 99
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

IMPLEMENTING REMOTE IMAGE CAPTURE/CONTROL IN A WIRELESS SENSOR NETWORK UTILIZING THE IEEE 802.15.4 STANDARD

Daniel E. Krehling
Captain, United States Marine Corps
B.S., University of South Alabama, 2001

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2009

Author: Daniel E. Krehling

Approved by: John Gibson
Thesis Co-Advisor

Gurminder Singh
Thesis Co-Advisor

Peter Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Today's warfighter requires an in-d epth view of the battle space in order to best plan for future operations, assess the current operating enviro nment, and prevent or respond to attacks. The deployment and use of wireless sensor devices could serve as a force multiplier by enhancing a commander's security posture, providing a view of the current environment, and gathering intelligence for analysis. The use of low-power imaging devices, coupled with the flexibility provided by a wireless sensor network, could provide such enhancements.

The objective of this research was to explore the feasibility of remote management and control of a low-power/low-cost wireless sensor network by implementing a point-to-point wireless network utilizing IEEE 802.15.4 equipped devices to control, capture, and transfer image data from a remote sensor node to the controlling host. This platform was used to test the viability of the system at various ranges and operating environments. The results demonstrated that the IEEE 802.15.4 compliant devices used in this research are able to operate over long distances (1000 meters), in harsh RF environments, with a high degree of reliability.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The navigation of this difficult task would not have been possible without the help, guidance, and moral support of faculty, friends, and most importantly, my family. To Professors Gurminder Singh and John Gibson, thanks for giving me the opportunity to investigate a topic for which I possess a great deal of interest. The journey has been challenging, enlightening, and above all, rewarding. Furthermore, thank you for your time, knowledge, and consideration in the development of this thesis.

To my family, especially my wife, Cookie, thank you for your support, inspiration, and most of all, your understanding in the accomplishment of this task. The late nights, short weekends, and weeks apart while I worked towards my goal are a testament of your love and devotion. Finally, your tolerance of an impatient boss while marking 100 meter increments with a GPS in one hand, a cell phone against your ear, and a XBee radio on a pole in your other hand, demonstrates your can-do mentality and willingness to get the job done. I am forever grateful.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

The past decade has seen an explosion in the research, development, and deployment of devices for use in wireless sensor networks (WSN). In the February 2003 edition of Technology Review, WSNs were listed as one of the top ten emerging technologies poised to dramatically influence the field of computing [1]. These networks include both the sensors that detect aspects of the physical environment, as well as the devices that perform the routing of the captured information in the network to the desired destination. WSNs have been around for many years, however, renewed interest has resulted from the popularity of wireless local area networks and their underlying technologies, such as 802.11(a), (b), and (g). Advances in integrated circuit design and microprocessor technologies, along with advanced routing algorithms and low-power operation, have propelled WSNs into the twenty-first century. These devices are quickly becoming a ubiquitous part of our environment as the cost and power requirements continue to fall and the processing and overall functionality continue to rise. WSNs can be found in a myriad of places, including industrial and home automation, workplace safety, consumer electronics, healthcare monitoring, agricultural control, and military applications [2].

The military as a whole relies on actionable, reliable information for the formulation and conduct of their concept of operations. The deployment and use of wireless sensor devices could serve as a force multiplier by enhancing a commander's security posture, providing a view of the current operating environment, and gathering intelligence for analysis. The use of low-power imaging devices coupled with the flexibility provided by a wireless sensor network could provide such enhancements.

B. PROBLEM DESCRIPTION

Wireless sensor networks allow us to monitor, evaluate, and control aspects of our environment. They allow us to capture portions of our physical world that in the past were impossible or difficult at best to detect and monitor without highly specialized and costly equipment. Aspects of the environment that are desirable to monitor include: magnetic, seismic, acoustic, motion, and imagery just to name a few [3]. In the realm of wireless sensor networks, small devices that can operate on relatively low bandwidth levels and small amounts of power for extended periods of time with little or no maintenance is a key characteristic. A large majority of the sensors available for use today are easily geared to match these requirements. However, unlike many of the other sensors that can be deployed, remote imaging can weigh heavily on the resources available in a sensor network. Image sensors tend to consume large amounts of power with an even greater need for bandwidth to transfer the smallest of image data. Fortunately, with advances in semiconductor technology and compression algorithms, low-power and low-bandwidth image sensors are becoming a reality. This capability combined with the low-power, low-data rate, ad hoc wireless nodes allows a developer to create robust remote image sensing applications that can be easily deployed and utilized. This opens the door to the following hypothesis and underlying topic of this thesis research:

Use of IEEE 802.15.4 enabled devices, coupled with image compression techniques at the sensor node, allow for efficient image capture and device control in a wireless sensor network.

C. THESIS ORGANIZATION

Chapter II of this thesis provides a broad overview of the early history of wireless sensor networks. It also discusses the recent applications of WSNs and the renewed interest in the technologies that led to the creation of the IEEE 802.15.4 standard. The discussion continues with a detailed description of the standard itself and concludes with current and future uses of the technology.

Chapter III explores the technical approach used to enable the control of and image capture by an imaging sensor via the use of IEEE 802.15.4 enabled devices. This chapter begins with a discussion of the individual hardware components and ends with a description of the software components that were developed to enable the desired functionality.

Chapter IV describes the results of the overall system architecture and its operation, as well as the lessons learned in the process. Likewise, the results of range and reliability testing are provided as proof of concept.

Chapter V surveys the entire area of study. Conclusions of the proposed hypothesis, system design and approach are discussed and followed by a description of future work that could further propel the technology.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. WSN ORIGIN

Wireless sensor networks were conceived and implemented in the late seventies, as a result of military research and development at Defense Advanced Research Projects Agency (DARPA). As shown in Figure 1 a wireless sensor network is a wireless network consisting of spatially distributed autonomous devices that utilize sensing, computing, and communication to cooperatively monitor and report on conditions in the surrounding environment [4, 5].

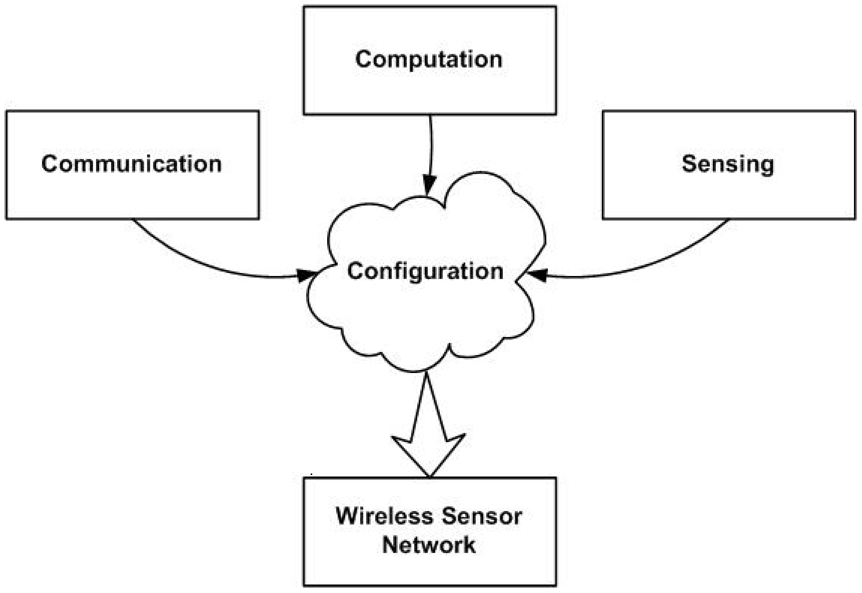


Figure 1 WSN Composition

As such, one of the earliest predecessors to the (WSN) came into existence during the Cold War and was coined the Sound Surveillance System (SOSUS). SOSUS was a system of underwater acoustic sensors that were placed at strategic locations to detect, track, and monitor Soviet submarines. The information gathered by the string of sensor arrays was sent to an individual collection node, such as NAVFAC Point Sur south of Monterey, CA, where the data was processed and observed by specialists who were trained to identify the presence of a submarine in the output data. Pertinent information was then relayed to higher echelons. The system is still in use today; however, its mission has changed from military use to an environmental role in monitoring oceanic events for the National Oceanographic and Atmospheric Administration [6].

DARPA continued to pursue its research in the sensor network field in the early eighties with the launch of the Distributed Sensor Networks (DSN) program. Several participating research laboratories, including MIT Lincoln, and Carnegie Mellon University (CMU), set out to test the plausibility of incorporating the communications techniques used in the Arpanet (predecessor of the Internet) to sensor networks themselves. As a result, MIT Lincoln developed a wireless sensor network prototype that tracked low-flying aircraft by the use of acoustic sensors. The sensor array consisted of nine microphones arranged in concentric triangles and was attached to a PDP11/34 computer for signal processing. Target processing at the node consisted of three MC68000 processors with 256kB memory, 512kB shared memory, and a custom operating system. Communication to the other participating nodes was achieved by Ethernet and microwave radio [7] . The equipment used to create the WSN components (acoustic array, mobile node, equipment rack) in Figure 2 consisted of custom-built components. The size, as well as the cost of these components, was quite large but typical for such state of the art equipment of the time.

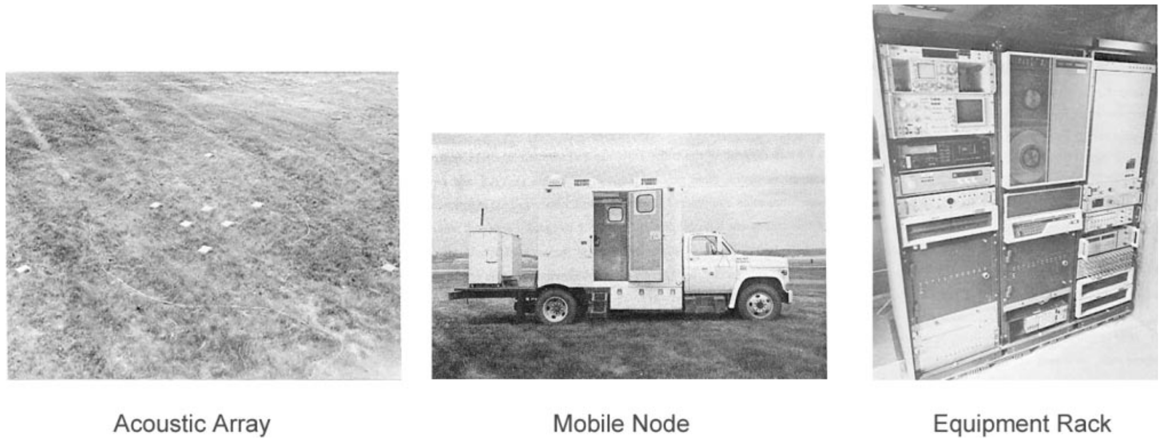


Figure 2 Sensor Network Components [From:[7]]

The system was one of the first to adhere to the defined WSN principles of sensing, computation, and (wireless) communication. The signal s from the acoustic array were sent to the mobile node where they were processed and ultimately broadcast to other participating nodes for their tracking and processing needs.

B. RECENT USES / APPLICATIONS

The past decade and a half has yielded a number of advances in the WSN field and as such has enabled the technology to blossom across a wide gamut of applications. While military applications dominated the space and fueled continued research, WSNs were soon expanded to environmental, industrial, agricultural, and home automation applications.

1. Glacier Monitoring

One such implementation of a modern day WSN was the use of the technology to monitor and report on the sub-glacier environments in Briksdalsbreen, Norway in order to gain a better understanding of the effects of climate change [8]. The underlying problem space represents a perfect use-case for the deployment of WSN technology; devices need to be deployed in harsh operating environments, for long periods of time, on a finite source of energy, with a low-level data rate. Figure 3 illustrates the system configuration where the sensor devices in Figure 4 are placed in holes drilled into the glacier and the earth below the ice.

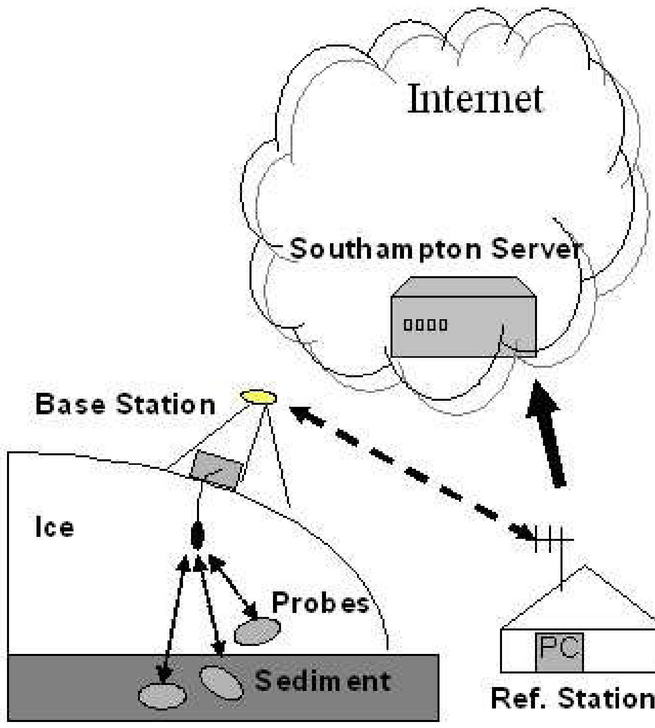


Figure 3   Glacsweb System Overview [From:[8]]

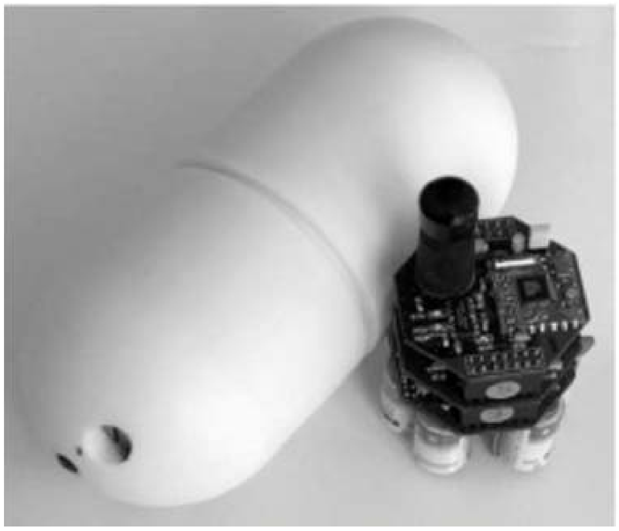


Figure 4 Probe Sensors [From:[8]]

When commanded, the sensors measure pressure, temperature, and orientation of the probe and report this information to a central base station at the top of the glacier. The sensor probe itself is comprised of a 173 MHz transceiver, a PIC based microcontroller for control and signal processing, analog components for sensor input interfacing, the temperature, pressure, and position sensors, and six lithium thionyl chloride cells for the energy source; a great amount of functionality housed in a container the size of a palm-sized stone.

Communication between the sensors and the base station was performed via the use of the 173 MHz transceiver as the low-level radio hardware. Considering the possibility of changing radio hardware, the design team opted for a custom built communications protocol. The protocol for the system was based on a master-slave approach in order to reduce complexity, collisions, and power consumption while avoiding the use of MAC protocols. The sensor probes remained in a passive state until polled by the base station at the top of the glacier.

The base station consisted of an ARM-based, single-board computer (BitsyX, an embedded Linux machine), GPS, weather instruments, radio modem, and lead acid batteries refreshed by a wind generator and solar panels. The base station subsequently communicates the retrieved probe information via a wireless point-to-point (PPP) link to the main collection facility. The base station also serves as a gateway to the probe's firmware. Users are able to log into the base station via telnet and make changes to the base station and probe's operating parameters and user space scripts.

2. Grape Vineyard Monitoring

Another well-defined problem space that is well-suited for the deployment of a WSN is in the use of the technology to monitor the environmental aspects of a grape vineyard [9]. Researchers wanted to determine the viability of ubiquitous computing in the agricultural domain. The site of the WSN application was located in a vineyard where the sensor nodes would measure and report on temperature variations. The gathered data would then be used to determine the optimum period for grape harvesting, the prevention of potential diseases (powdery mildew), as well as the prevention of crop damage due to the rapid onset of low temperatures.

The system architecture, as depicted in Figure 5 , is a preconfigured (known routes), multi-hop WSN consisting of a gateway node, sixteen backbone nodes, and three leaf nodes per backbone node for a total of sixty-five nodes.

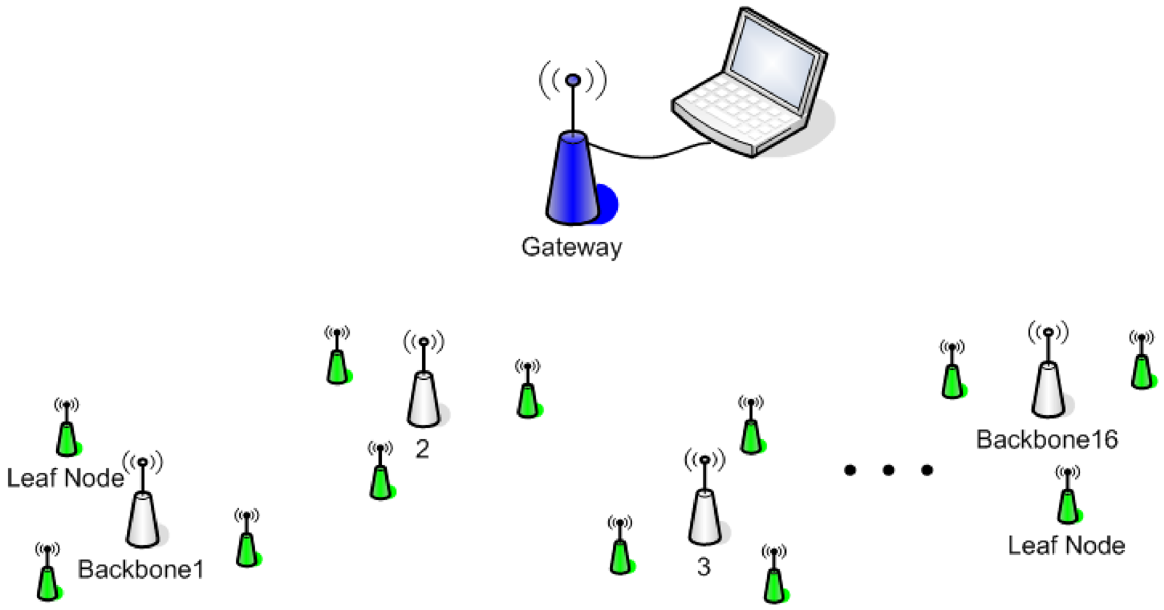


Figure 5 Vineyard WSN Architecture

The backbone and leaf nodes in Figure 6 are spread strategically over a two-acre area of the vineyard. Leaf nodes are typically no more than fifteen meters from their respective backbone node, while each backbone node is within twenty-five meters of its neighboring backbone node.

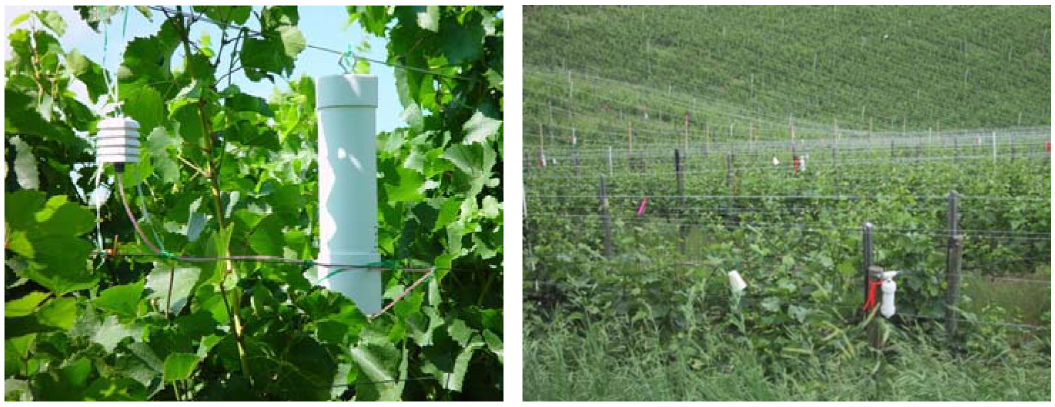


Figure 6 Vineyard Sensor Nodes [From:[10]]

The network operates based on synchronization signals sent out from the gateway to the backbone, and subsequently, the leaf nodes. After synchronization, each leaf node reports its data to the respective backbone node during its prescribed time slot. Each backbone node then sends the collected data to two (redundancy) of the upstream nodes (maximum of eight hops) until the data reaches the gateway. When the data transfer is complete, the leaf, as well as the backbone nodes, go back to sleep until the next cycle, which occurs every five minutes.

The design and deployment of the WSN in this particular regime allowed the researchers to collect relatively high resolution data from the two acre test site. The results from the data shattered previously accepted heuristics that had been in place to help determine crop maturation and disease incubation; namely, that elevation was a predictor of temperature and a two-acre block was assumed to have similar growing conditions. However, the findings showed that temperature hotspots across the vineyard varied from day-to-day. Furthermore, the elevation, as per the diurnal data, was not a valid predictor of temperature.

The WSNs of late were subject to a number of characteristics that hampered their performance and, therefore, their viability and adoptability into mainstream computing. Rudimentary power-management schemes or the lack there of, bulky and unintegrated electrical components, and nonstandardized communication protocols that were often overkill for the size and scope of the application, or had to be custom-built, plagued early system integration.

While Moore’s law was chipping away at several of the aforementioned issues by shrinking the size and power requirements of the WSN devices, the lack of a viable communications standard for WSNs in general sparked the debate for the development of such a standard in 2000; enter the 802.15 Working Group.

C. THE IEEE 802.15.4 STANDARD

The 802.15 Working Group was sanctioned in March 1999 in an effort to design a standard for wireless personal area networks (WPAN). The working group was focused on the creation of the 802.15.1 standard that was based on Bluetooth technology. However, during their collaboration members of the group soon realized that their dream of ubiquitous, untethered, short-range communications could be further divided into three distinct standards: the original 802.15.1 standard, 802.15.3 for high-speed media devices, and the IEEE 802.15.4 standard for low-rate WPANs. As such, the IEEE 802.15.4 Task Group was commissioned in December 2000 and tasked to define a standard that would provide ultra-low complexity, low-cost, and low-power wireless communications for inexpensive fixed, portable, and moving devices [11].

Several years after the commissioning of the working group, the IEEE 802.15.4 standard was finally published in October of 2003. The new standard defined what would become known as low-rate wir eless personal area networks (LR-WPAN) where applications could be developed to function with low-data rates and relaxed latency and quality of service (QOS) requirements. Moreover, the standard was designed with simplicity in mind and subscribed to the premise of low-power functionality. Figure 7 highlights the standard's power, data rate, and complexity characteristics as compared to the other wireless standards.

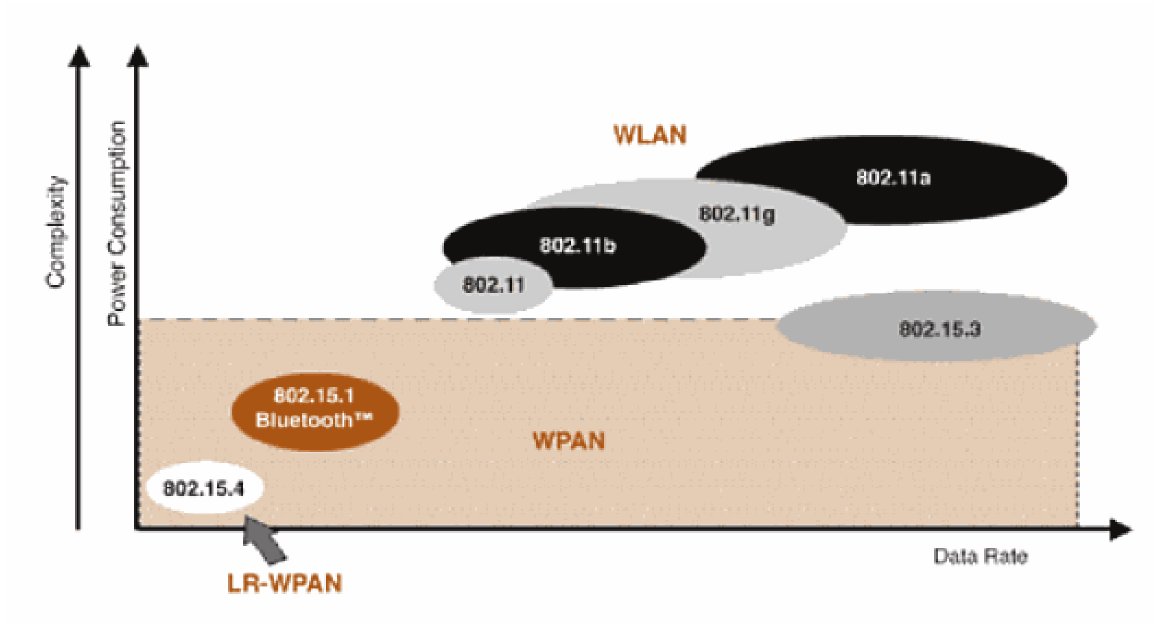
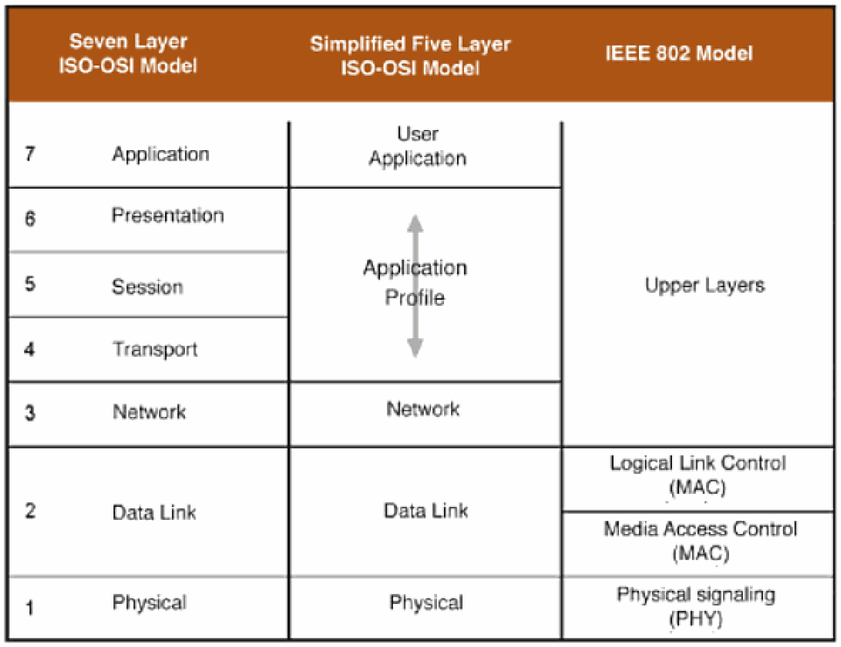


Figure 7 Wireless Standard Comparison [From:[11]]

The wireless protocol standard defines the lower two layers (Data Link and Physical) of the traditional Open System Interconnection (OSI) reference model: the medium access control (MAC) and the physical (PHY) layers. The upper layers (Network, Transport, etc.) are not defined by the standard and thus are implementation independent. However, other standards such as the ZigBee standard have been developed to provide added functionality at the upper layers to include security and ad-hoc routing mechanisms. Figure 8 shows the standard as it relates to the seven layer OSI model.



Seven Layer ISO-OSI Model		Simplified Five Layer ISO-OSI Model	IEEE 802 Model
7	Application	User Application	Upper Layers
6	Presentation	Application Profile	
5	Session		
4	Transport		
3	Network	Network	
2	Data Link	Data Link	Logical Link Control (MAC)
			Media Access Control (MAC)
1	Physical	Physical	Physical signaling (PHY)

Figure 8   Seven-Layer OSI vs IEEE 802 Model [From:[11]]

In general, the IEEE 802.15.4 standard sets forth the operation of two types of devices: full function (FFD) and reduced function (RFD) [12]. Full function devices are able to operate over the full range of MAC layer services allowing them to function as network coordinators, as well as network devices. Conversely, the reduced function devices operate over a restricted set of the MAC layer services, thus, allowing them to maximize power savings and minimize complexity. The use of these two device types allows for the creation of two possible network topologies: star and peer-to-peer, as shown in Figure 9 The coordinator, a FFD, is responsible for the creation and maintenance of a given network where other FFDs and RFDs are allowed to join and depart the network. Moreover, the coordinator can serve as a gateway to an external host, or more importantly, as a bridge to a disparate network such as the Internet.

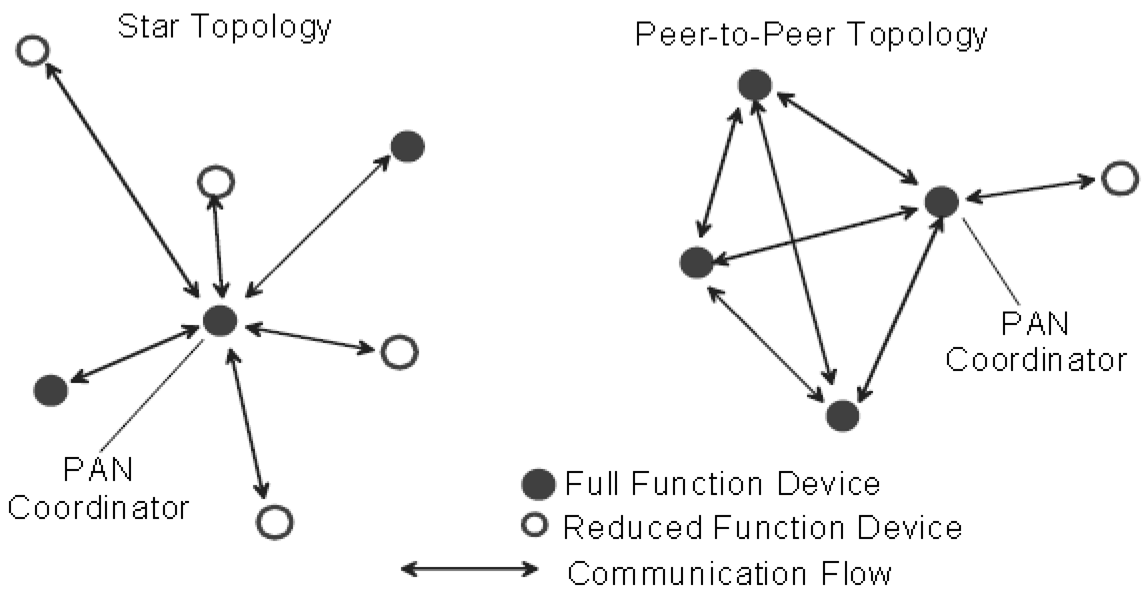


Figure 9 Star and Peer-To-Peer Topology [From:[12]]

RFDs communicate only with their respective network coordinator while FFDs can communicate with other in-range FFDs in the network, as well as with the PAN coordinator. Finally, the standard specifies the three working frequency bands, 868MHz, 915MHz, and the 2.4GHz industrial scientific and medical (ISM) band, as well as the respective data rates achievable in each band.

1. PHY Layer

The physical layer is composed of the three aforementioned frequency bands that are subsequently grouped into the following categories: low-band and high-band. The 868/915MHz bands fall into the low-band category, while the 2.4GHz band is considered high-band.

Both categories use direct sequence spread spectrum (DSSS) for digital signal transmission but separate modulation techniques; the low-band uses binary phase-shift keying (BPSK) and the high-band uses offset quadrature phase-shift keying (O-QPSK). The bit rates provided by the three bands are as follows: 20 Kbps for 868 MHz over a single channel, 40 Kbps for 915 MHz over ten channels and 250 Kbps for 2.4GHz over sixteen possible channels [12]. The MAC layer makes use of only a single channel at a time, as it is not a multichannel protocol. Table 1 summarizes the highlights of the PHY layer frequencies while Figure 10 provides a graphical representation of the 27 channels for the high and low bands defined in the standard.

Band	Frequency Band	Bit Rate	Symbol Rate	DSSS Spreading Parameters	
				Modulation	Chip Rate
868 MHz	868-868.6 MHz	20 Kbps	20 ksymbols/s	Binary Phase Shift Keying (BPSK)	300 Kchips/s
915 MHz	902-928 MHz	40 Kbps	40 ksymbols/s	Binary Phase Shift Keying (BPSK)	600 Kchips/s
2.4 GHz	2.4-2.4835 GHz	250 Kbps	62.5 ksymbols/s	Offset Quadrature Phase Shift Keying (O-QPSK)	2 Mchips/s

Table 1 IEEE 802.15.4 Frequency Band and Modulation Parameters [From:[11]]

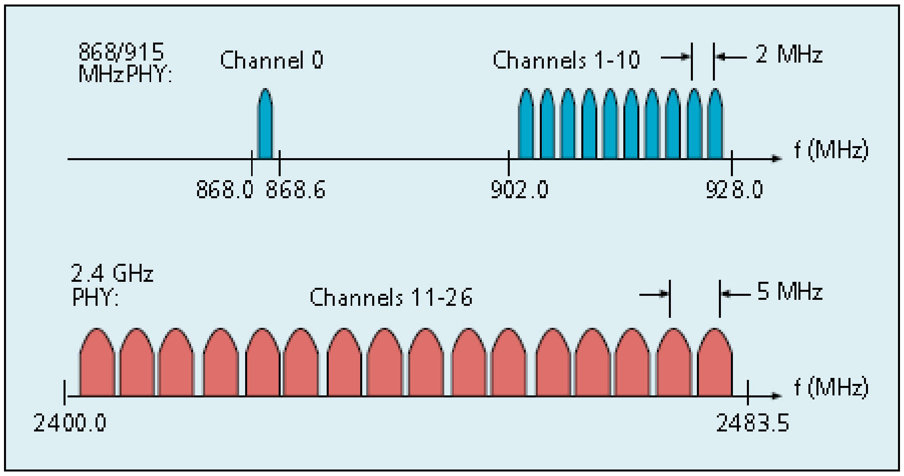


Figure 10 IEEE 802.15.4 Frequency Channels [From:[13]]

The PHY layer, in general terms, is responsible for the establishment of RF communications between devices, the modulation and demodulation of the bit stream, transmitter/receiver synchronization, and packet level synchronization. The PHY layer provides this functionality through six delineated services and their underlying primitives or functions. The six services are as follows: Transceiver Activation/Deactivation, Energy Detection, Link Quality Indication, Channel Selection, Clear Channel Assessment, and the Transmission and Reception of data packets [12].

- Transceiver Activation/Deactivation: this service is responsible for the actual power-up / power-down of the transceiver. This action allows for low-power operation by controlling the duty cycle.
- Energy Detection (ED): ED is used to estimate the RF energy that exists in the environment on the current channel. This service is intended to be

used by a network layer (undefined upper layer) as part of a channel selection algorithm.

- Link Quality Indication (LQI): LQI is an assessment of the quality of a received packet that is provided to the MAC sublayer for every received packet. The LQI is provided for use as a service to upper layers, such as network or application, which are not defined by this standard.
- Channel Selection: this service is provided as a means to select the individual channel that will be used for operation as commanded by the MAC layer.
- Clear Channel Assessment (CCA): CCA is the service that is used by the CSMA/CA protocol to check for competing devices that are transmitting on the same operating frequency. The service enables the receiver, monitors the medium, disables the receiver, and finally, reports the status of the medium to the MAC layer.
- Transmission and Reception of data packets: this service is responsible for the encapsulation of the MAC sublayer frame withi n the PHY Protocol Data Unit (PPDU), shown in Figure 11 and the subsequent transmission of the PPDU over the wireless medium. Conversely, this service is also responsible for the reception of raw bits from the medium, the packaging of the bits into the PPDU frame structure, and making the

encapsulated data available to the MAC layer for extraction. Figure 12 highlights the encapsulation of the MAC sublayer frame within the PPDU.

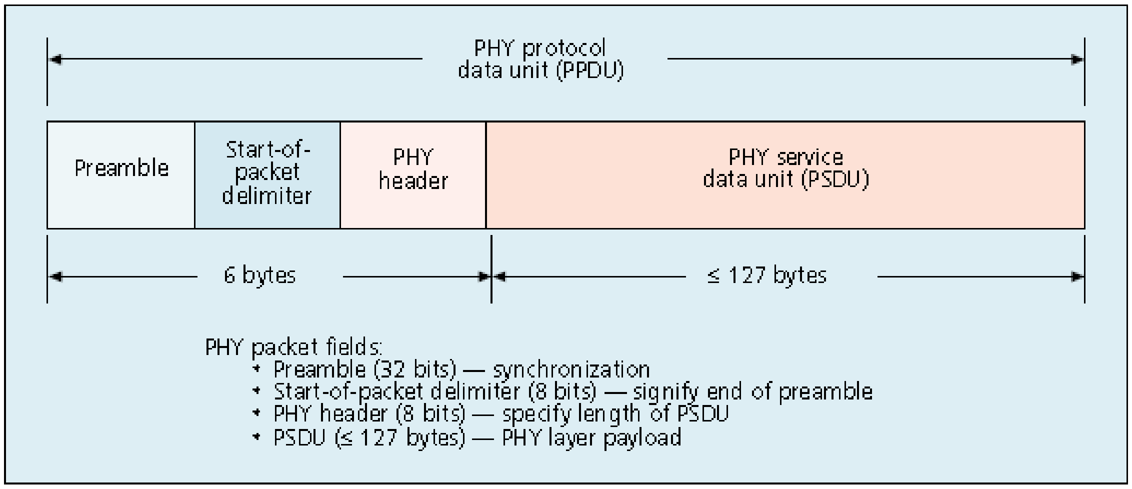

PHY protocol data unit (PPDU)
Preamble
Start-of-packet delimiter
PHY header
PHY service data unit (PSDU)
6 bytes
≤ 127 bytes
PHY packet fields:
• Preamble (32 bits) — synchronization
• Start-of-packet delimiter (8 bits) — signify end of preamble
• PHY header (8 bits) — specify length of PSDU
• PSDU (≤ 127 bytes) — PHY layer payload

Figure 11 IEEE 802.15.4 PHY Protocol Data Unit [From:[13]]

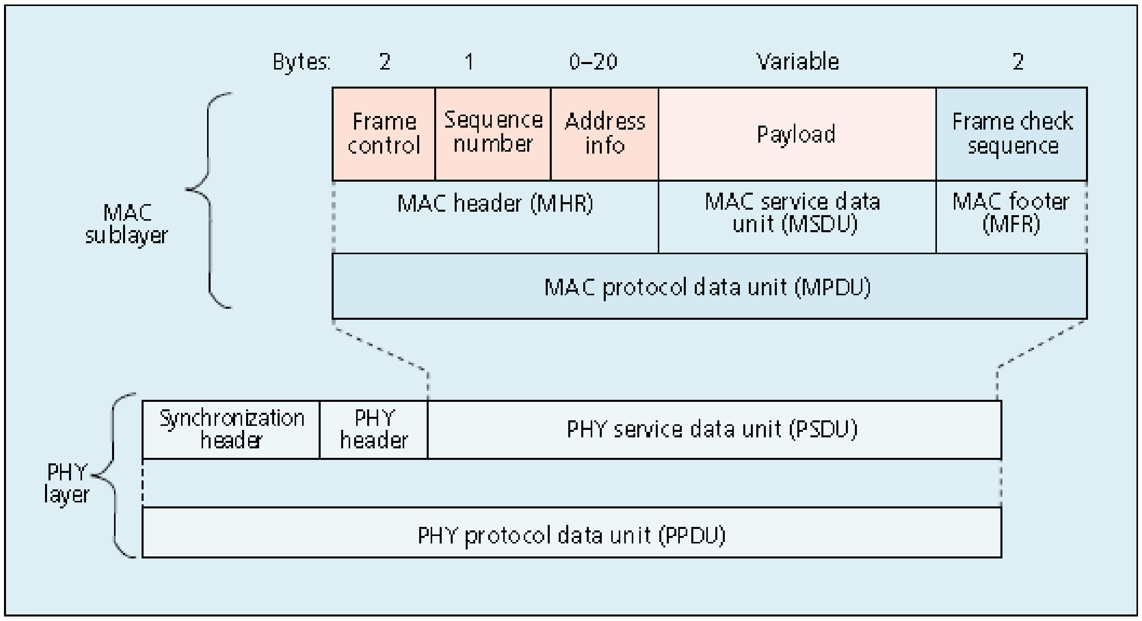


Figure 12 MAC and PHY Frame Structure [From:[13]]

The IEEE 802.15.4 standard also defines some of the operating characteristics as they apply to the radio transceiver at the PHY layer to include output power, receiver sensitivity, and range [11, 13]. In regards to output power, the standard allows for a wide power range that is bound on the lower side by a minimum transmitter output of -3 dBm (.5 mW). The upper bound is governed by the regulatory agency for the country where the device is in use. For receiver sensitivity, the standard states that the device must be able to correctly decode a signal with an input power of -85 dBm or more for the high band (2.4 GHz), while the low-band receiver's sensitivity is set at -92 dBm or more. The range is a factor of the output power and the receiver sensitivity, as well as the type of antenna in use and the overall cleanliness of the RF operating environment. That being said, a 1 mW device should be expected to operate over a range of ten to twenty meters. However, commercial devices do exist that have power output of up to 100 mW and receiver sensitivities that exceed those specified in the standard. Such devices could be expected to operate at ranges well over 1000 meters.

2. MAC Layer

The MAC sublayer of the IEEE 802.15.4 specification, along with the IEEE 802.2 logical link control (LLC) sublayer, comprise the Link layer equivalent of the ISO OSI seven-layer model. In general terms, the MAC itself provides reliable data delivery and access to the shared wireless medium in the PHY layer [11]. These two generalized services provided by the MAC are further expanded into several functions to include the following: beacon management, channel access, guaranteed time slot management, frame validation, acknowledged frame delivery, association/disassociation, and a set of hooks for implementing security at the upper layers [12]. The MAC provides the mechanisms by which simple, yet efficient LR-WPAN network topologies can be constructed and controlled, most notably star and peer-to-peer configurations. More advanced topologies (mesh, multi-hop) are possible but require implementation in the upper layers, as they are not defined in this standard.

Along with the flexibility provided by the various topologies, the MAC layer also allows for two operating schemes in the organization and deployment of the FFD and RFDs in the network: beacon mode and nonbeacon mode. These two modes define the operating characteristics of the established LR-WPAN in terms of quality of service (QOS), latency, and power efficiency.

In the beacon mode of operation, the PAN coordinator provides a means of synchronization to the other participating devices in the network by means of a time specific beacon in what has been coined a Superframe, as shown in Figure 13 As seen in the diagram, the Superframe is divided into active, as well as inactive blocks that are bound by the actual beacon time slots. Simply stated, the inactive period (optional depending on configuration) is a time slot where network devices including the coordinator can turn off their transceivers and enter a low-power state. The only caveat is that these devices must wake just prior to the start of the next beacon in order to maintain their synchronization.

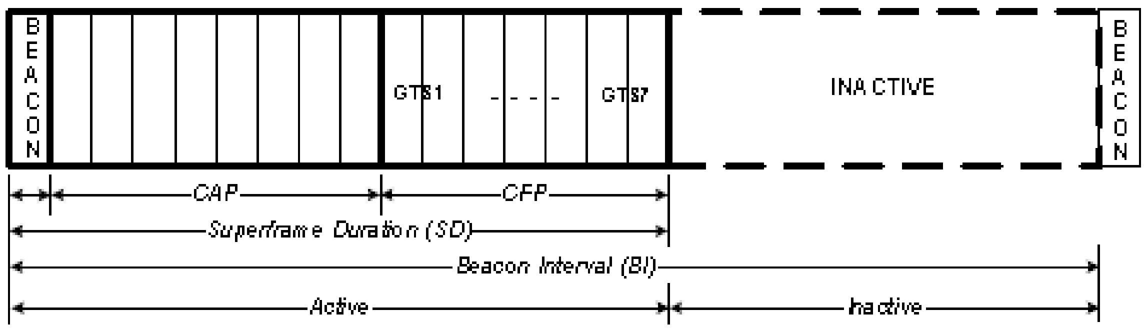


Figure 13 MAC layer Superframe [From:[14]]

Conversely, the active period is composed of a Contention Access Period (CAP) followed by an optional Contention Free Period (CFP) [14]. The CAP is a programmable period of time between bounding beacons where devices can perform such functions as request to join the network, poll the coordinator for queued data, and exchange data with the coordinator. Communication during the CAP is performed via a slotted CSMA-CA mechanism. Since the devices in the beacon enabled mode are synchronized, the standard enhances the CSMA-CA protocol in use during the CAP by dividing the CAP into time slots called backoff periods. Each backoff period is equivalent to twenty channel symbol times. These backoff periods in the CAP are the slots used by the slotted CSMA-CA protocol. When a device desires to transmit a frame, it waits for the next slot boundary and draws a random integer r. The random integer is bound by interval $[0,2^{macMinBE}-1]$, where macMinBE is the minimum backoff exponent with a range of zero to eight and a default value of three. The device waits for r backoff periods and checks the medium and if idle, waits for the next backoff period and senses the medium again. If both checks show the medium as idle, the device concludes that it has won contention and starts the data transmission. On the other hand, if the channel was busy in any of the sensing operations, the backoff exponent and the number of backoffs that have occurred are incremented. If the number of backoffs is below the threshold, the device repeats the operation. However, if the threshold is passed, the device drops the frame and declares a failure. Devices that choose to communicate during the CAP must complete the exchange prior to the beginning of the optional CFP or the beginning of the next beacon in the case that a CFP is not in use.

However, the configuration of the network may be such that a device in the topology needs a specific amount of bandwidth or lower level of latency. In such cases, the MAC can be configured to provide Guaranteed Time Slots (GTS) for those devices requesting such service [12]. The GTS, up to a maximum of seven, can be provided by the coordinator upon request and collectively form the CFP. The GTSs are fixed in size and governed by the coordinator in both the allocation and de-allocation of the time slot. Moreover, any device wishing to utilize the GTS mechanism requests the service during the CAP. The coordinator, upon approval, sends an acknowledgement to the device and subsequently sends the GTS parameters to the device in the follow-on beacon frames. These parameters include the specific time slot given to the device, as well as the number of contiguous time slots awarded; a device can request more than one GTS. Devices that are allowed to use the GTS must ensure that they have completed their exchange prior to the end of their GTS or the end of the CFP.

Unlike the operation of the Beacon mode, the Nonbeacon Mode eliminates the use of the beacon and the Superframe mechanisms. Instead, this simplified mode relates closely to the operation of the aforementioned CAP where devices compete openly for use of the medium with the only difference being the use of an un-slotted CSMA-CA protocol for the detection of competing nodes. If upon successful testing of the medium, the device simply infers success and immediately begins to communicate with the destination device. One of the major drawbacks to this configuration is that the coordinator must remain switched on constantly to monitor requests from joining or departing nodes as well as communication requests from currently configured devices. End devices, however, can follow their own sleep cycle, thus preserving the low-power premise of the standard.

Central to the functionality provided by the MAC layer is the MAC protocol data unit (MPDU). The MPDU is the frame structure that encapsulates the payload data from the upper layers and provides a set of fields that are used to facilitate such functions as addressing, acknowledgements, security, and error correction. A generic MPDU is composed of three sections: the MAC header (MHR), the MAC service data unit (MSDU), and the MAC footer (MFR). All three sections can be further divided depending on the type of frame in use. Ultimately, the generic MPDU, shown in Figure 14 is passed to the PHY layer where it is packaged inside the PHY protocol data unit and sent out over the wireless medium.

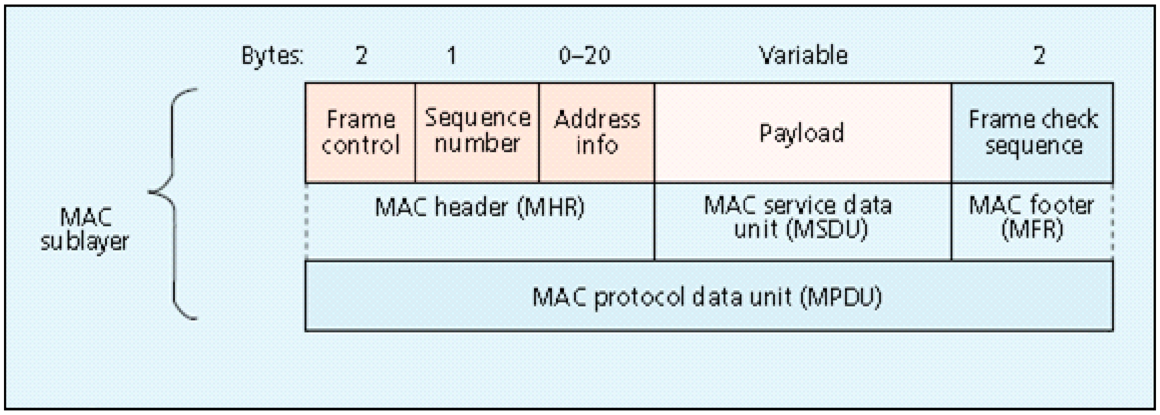


Figure 14 IEEE 802.15.4 MPDU [After:[13]]

The standard defines four distinct frame types for the MAC sublayer: beacon frame, data frame, acknowledgment frame, and MAC command frame [12]. The beacon and data frames contain information from the upper layers while the acknowledgment and MAC command frames are generated by the MAC itself. The MHR contains the information that inherently defines the remainder of the frame, especially the type of frame to be used. This section contains information regarding acknowledgements, as well as the addressing information and scheme in use (16 or 64 bit address format). The MSDU is a variable length field that contains the data from the upper layers or data generated by the MAC sublayer itself in the case of a MAC Command frame. However, the standard defines the maximum length of a complete MAC frame to be 127 bytes. The final section of every frame type is the frame footer, MFR. This section is a sixteen bit number called the frame check sequence (FCS) and is an International Telecommunication Union – Telecommunication Standardization Sector (ITU-T) cyclic redundancy check (CRC) [11]. The FCS is used to detect errors in each and every frame.

The MAC sublayer with its simplified frame structure and small, efficient protocol definition provides reliable data communications through the use of a simple, full-handshake mechanism. The communications process is based on three distinct transaction types: data transfer to a coordinator where the end device transmits the data, data transfer from a coordinator where the end device receives the data, and data transfer between two peer devices. These three data transfer schemes are performed in both the Beacon and Nonbeacon modes of operation. Figure 15 (a) represents Beacon mode data exchange with data transfer from a coordinator and data transfer to a coordinator, respectively. Similarly, Figure 15 (b) depicts the Nonbeacon mode data exchange with data transfer from a coordinator followed by data transfer to a coordinator from the end device.

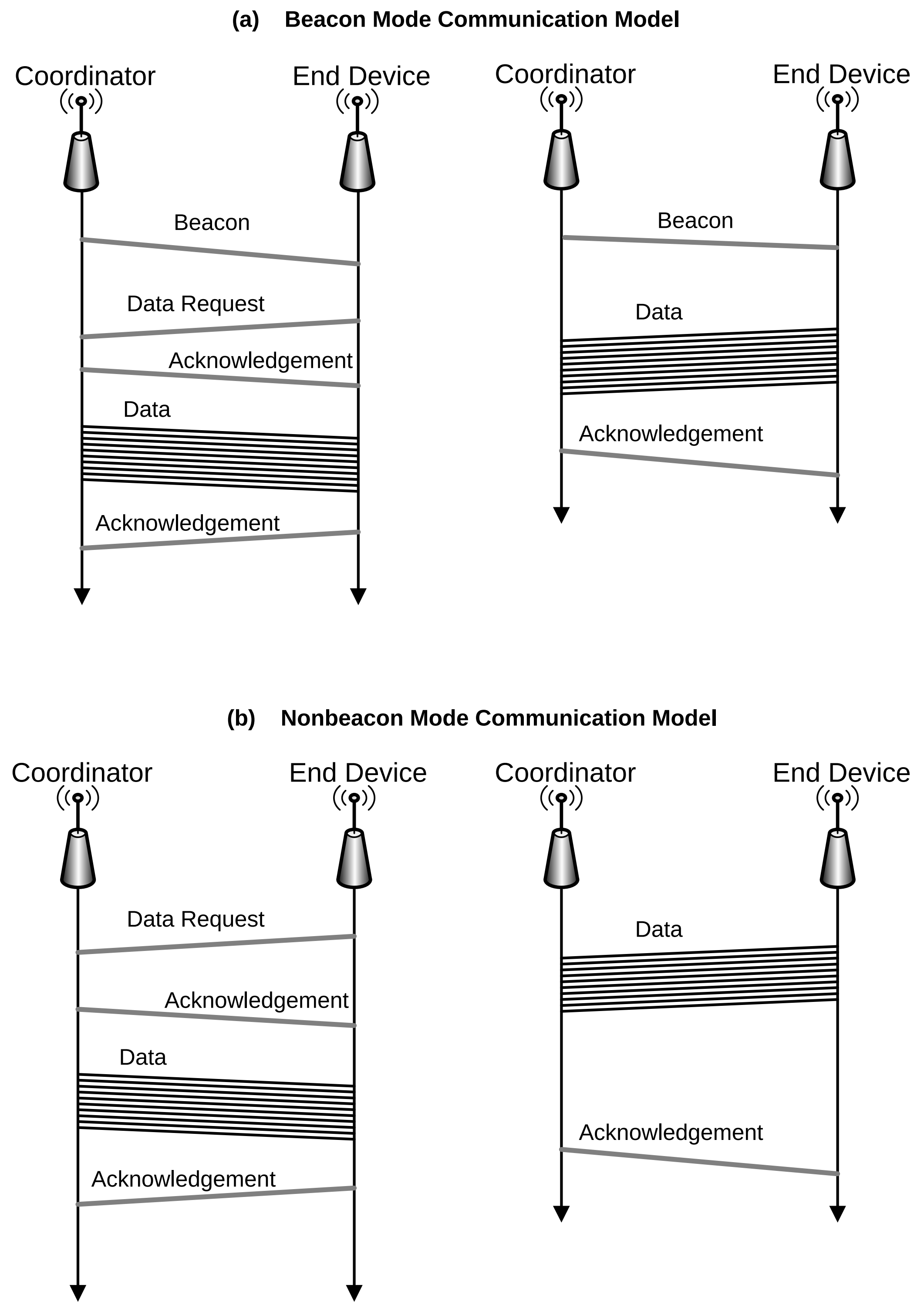


Figure 15 Star Topology Data Transfer Protocol for Beacon and Nonbeacon Modes

The reliable data communications protocol is made possible by the use of a simple acknowledgement scheme [12]. Acknowledgement frames, one of the four defined frame types, are sent immediately in response to data, as well as MAC command frames—Beacon and acknowledgement frames are never acknowledged. Since the acknowledgement frames are sent immediately following the associated frames, the CSMA mechanism is not employed. Acknowledgement frames are sent after successful reception and validation of the FCS field in the received frame and simply carry the frame sequence number of the successfully received frame. The originating device MAC looks for the arrival of an acknowledgement and continues with the next frame, resends the previous frame if no ACK was received, or ultimately terminates the transaction in the case of several failed transmissions.

3. IEEE 802.15.4 Summary

The goal of the IEEE 802.15.4 Task Group was to define a wireless standard that was simple, inexpensive, and would provide low-power / low-data rate communications to fixed, as well as mobile devices. The resulting standard of 2003, along with the refinements published in 2006, paved the way for low-rate personal area networks that have found their way into many aspects of the computing environment, to include environmental, industrial, home automation, and military applications. Moreover, due to advances in chip-manufacturing techniques, which have led to the production of a single, low-power, integ rated, wireless device, the standard is being propelled even further in applications such as parking lot management, vehicle-to-vehicle communications, humanitarian relief applications, and image capture and control.

By limiting the standard to the definition of only the MAC and PHY layers, the Task Group ensured the low-complexity premise, while allowing for the development of robust applications engineered in the upper layer of the OSI model. The low-power characteristic of the standard is achieved through mechanisms geared to control the duty cycle of the associated devices. For example, a coordinator and its connected devices could be configured in a manner where all participants sleep for a period of four minutes, wake to check for potential data exchange and return to the sleep state until the next iteration. Furthermore, a device that requires a fixed slice of bandwidth or a low-latency level can request and be configured to operate with some assurance through the use of a guaranteed time slot.

The standard was also developed with worldwide use in mind through the use of public-frequency bands, especially those in the 2.4 GHz ISM band. Therefore, by making use of multiple channels that employ robust modulation and spreading techniques, the standard ensures interoperability in environments where the medium is bombarded by competing technologies such as Bluetooth, 802.11, and cordless phones. The standard’s energy scan and clear channel assessment mechanisms allow a coordinator to dynamically switch the operating channel for the entire network to one that falls below some predetermined energy level.

III. TECHNICAL APPROACH

A. SYSTEM OVERVIEW

The underlying theme of this thesis research was to design, implement, and test the use of the IEEE 802.15.4 standard as an effective means to capture and control images in a wireless sensor network. The operable system would need to be able to send command information to a remote system and in turn receive and process the incoming data stream. Conversely, the remote end of the system would have to receive and parse the incoming system commands, execute as commanded, and subsequently provide image data, as well as correctly position the camera by way of a digital servo. System hardware, shown in Figure 16 included a Pocket PC PDA, a C328R JPEG camera, a digital servo, a servo controller, and most importantly, the IEEE 802.15.4 transceivers and their associated carrier boards.

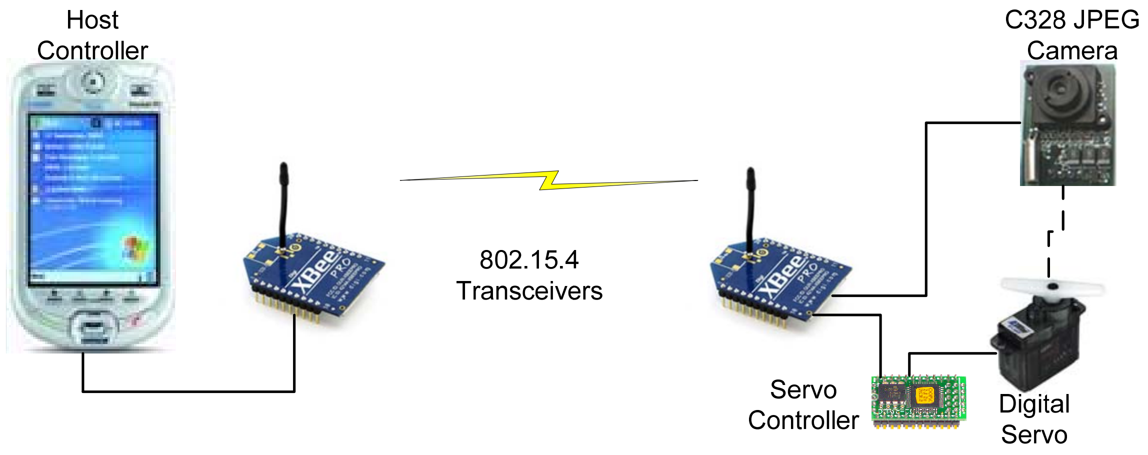


Figure 16 IEEE 802.15.4 Image Capture/Control System Diagram

B. HARDWARE DESCRIPTION AND DISCUSSION

1. Hewlett Packard hw6945 Pocket PC

The hw6945, displayed in Figure 17 is a Windows Mobile 5.0 device that has been incorporated into the system design and serves as the host controller for the WSN [15]. The PDA runs the software that allows the user to interact with the remote camera module over the IEEE 802.15.4 link and ultimately display the image on the TFT display.

Figure 17 Hewlett Packard hw6945 [From:[15]]

Specifications:

- Intel PXA270 Processor 416 MHz
- 64 MB SDRAM for running applications
- 45 MB persistent user storage
- 3.0” TFT Display (240x240 resolution)
- Lithium-ion 1200 mAh rechargeable battery
- RS232 Port for serial communications

2. OOPic Microcontroller

The OOPic, shown in Figure 18 is a PIC-based (Microchip Technology Inc.) device that provides a multitude of functions, particularly pulse width modulation (PWM) [16]. Programmable PWM is the mechanism by which the servo is positioned. The device is programmed by way of an integrated development environment that allows the user to develop the code in several different languages including C, Java, and Basic.

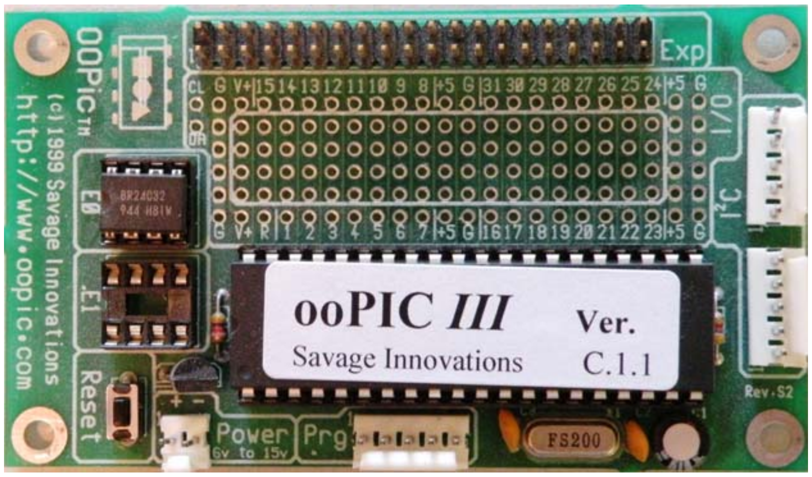


Figure 18 OOPIC Microcontroller

Specifications:

- 5 VDC operation
- 20 MHz clock (5 MHz effective) speed
- 31 General Purpose Input/Output Pins
- 4 Analog to Digital converters
- Asynchronous Serial Communications Port
- 2 Pulse width modulation channels

The OOPic microcontroller (MCU) is used solely to position the camera module as commanded by the user. In order to determine the position of the camera, the MCU's universal asynchronous receiver transmitter (UART) was configured to monitor the incoming byte stream for a predefined command sequence. Upon recognizing the command sequence, the MCU extracts the servo position information. Finally, the MCU uses the servo position information as an input parameter to the pulse width modulation object that subsequently drives the servo to the correct position.

3. E-flite Digital Servo

The E-flite digital servo, Figure 19 is a sub-micro servo device that operates at 5 VDC and provides 180 degrees of rotation [17]. It interfaces directly with the OOPIC MCU from which it receives the servo position signals.

Figure 19 E-flite Digital Servo

Specifications:

- 4.8 – 5.3 VDC operation
- Sub-micro, high-speed/high torque
- 17.2 oz/in of torque @ 4.8 VDC

- Digital operation
- .11 sec/60 deg @ 4.8 VDC travel speed

In the design of the system, the servo is used to slave the camera module through 180 degrees of rotation. The camera module is mechanically attached to the servo control platform. Pulse width modulated signals from the MCU are fed directly to the signal control line of the servo. The servo in turn drives the output shaft in accordance with the duty cycle of the input signal. Figure 20 provides a graphical representation of the PWM signals generated by the MCU as input to the digital servo.

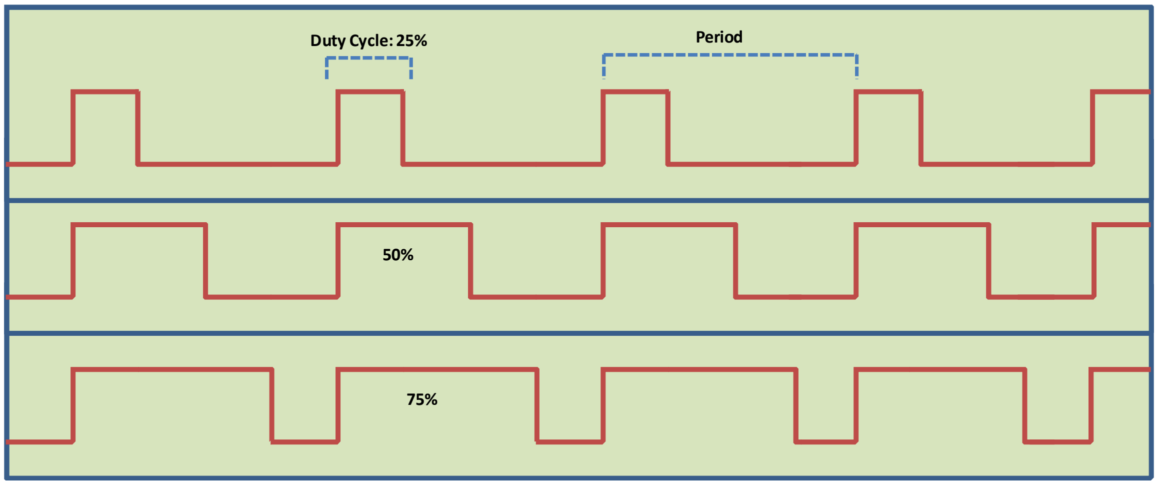


Figure 20 Generic Servo Control Signal

The primary characteristic that controls the position of the servo is the signal's duty cycle where duty cycle can be defined as:

duty cycle D = (on-time / period) * 100

From the diagram above, a duty cycle of 50 percent could represent the center position of the servo while duty cycles of 25 and 75 percent could represent the left and right limits of the servo, respectively.

4. C328R JPEG Camera

The strength of the C328R camera module and the reason for its use in the system design centers on the module's ability to capture and subsequently compress an image prior to transfer over the medium. The C328R, depicted in Figure 21 is a compact VGA camera mod ule that can perform as a standalone, JPEG-compressed, still camera [18]. It operates at 3.3 VDC and consumes a maximum of 60 mA. The device communicates with a host controller by way of a UART at rates of up to 115.2 Kbps and provides a set of user friendly commands for host control of the camera functions. JPEG images can be delivered at various resolutions: 80x64, 160x128, 320x240, and 640x480.

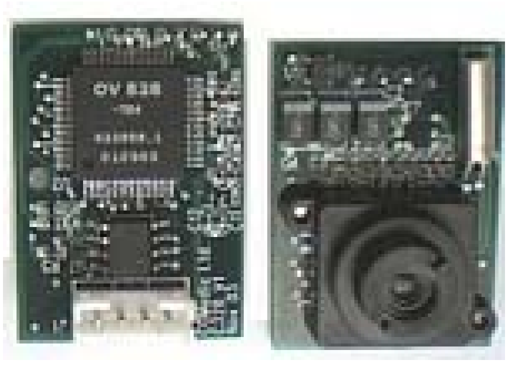


Figure 21 C328R JPEG Camera Module

Specifications:

- VGA resolution, down sample to QVGA or CIF
- 3.3 VDC operation
- 60 mA low-power consumption

- UART rates of up to 115.2 Kbps
- 100 µA standby power consumption
- Predefined commands for camera operation

The ability to control the image resolution is instrumental for the use of the technology in a wireless sensor network. As stated in the specifications, the module is controlled through the use of a series of predefined commands that are stored in the module’s onboard memory, as shown in Figure 22 Each command is six bytes in length and follows a structured format. These commands are issued by the host in a particular order and processed by the camera module. Confirmation of receipt between the host and camera are carried out via a series of ACK and NAK data packets. The general flow of events, Figure 23 that have to be performed in order to capture an image includes the following: Synchronize host to camera at 9600 baud, set the image resolution (JPEG at 320x240), set the size of the packets that will be used to return the image data (512 bytes), capture and store the image, and finally, retrieve image data from the camera module.

Command	ID Number	Parameter1	Parameter2	Parameter3	Parameter4
Initial	AA01h	00h	Color Type	RAW Resolution (Still image only)	JPEG Resolution
Get Picture	AA04h	Picture Type	00h	00h	00h
Snapshot	AA05h	Snapshot Type	Skip Frame Low Byte	Skip Frame High Byte	00h
Set Package Size	AA06h	08h	Package Size Low Byte	Package Size High Byte	00h
Set Baudrate	AA07h	1st Divider	2nd Divider	00h	00h
Reset	AA08h	Reset Type	00h	00h	xxh*
Power Off	AA09h	00h	00h	00h	00h
Data	AA0Ah	Data Type	Length Byte 0	Length Byte 1	Length Byte 2
SYNC	AA0Dh	00h	00h	00h	00h
ACK	AA0Eh	Command ID	ACK counter	00h / Package ID Byte 0	00h / Package ID Byte 1
NAK	AA0Fh	00h	NAK counter	Error Number	00h
Light Frequency	AA13h	Frequency Type	00h	00h	00h

Figure 22 C328R Command Set [From:[18]]

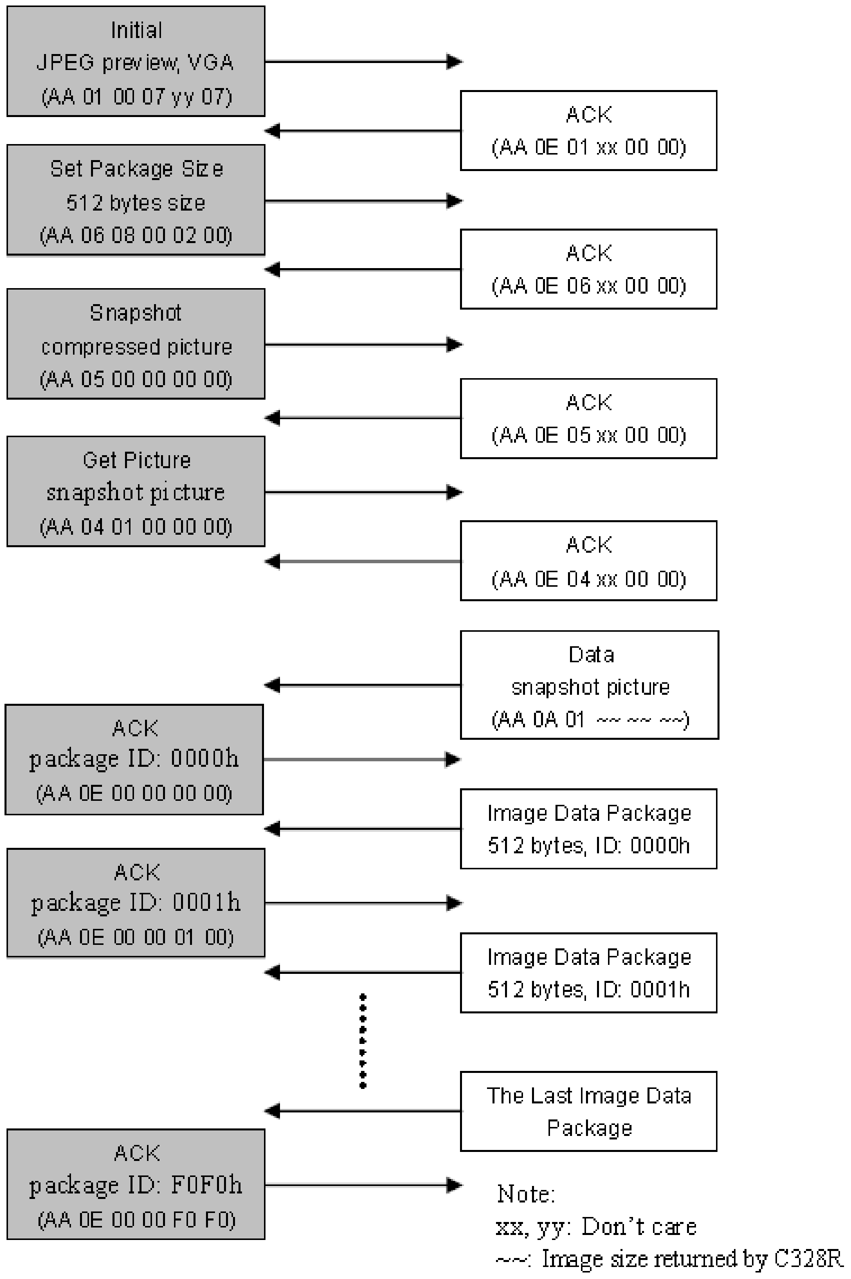


Figure 23 C328R Sequence of Operations [From:[18]]

5. XBee Pro IEEE 802.15.4 Transceiver

The XBee Pro transceiver, depicted in Figure 24 is an IEEE 802.15.4 compliant device engineered to meet the needs of low-cost, low-power wireless sensor networks [19]. The modules are instrumental in the development and deployment of peer-to-peer and point-to-multipoint applications. They provide a wide array of operating characteristics that enable the devices to be tailored to specific design constraints, such as low-power operation, operation in the 2.4 GHz ISM band, small form factor, and user control of the transceiver's operating parameters. Figure 25 highlights the module's compact design characteristics.

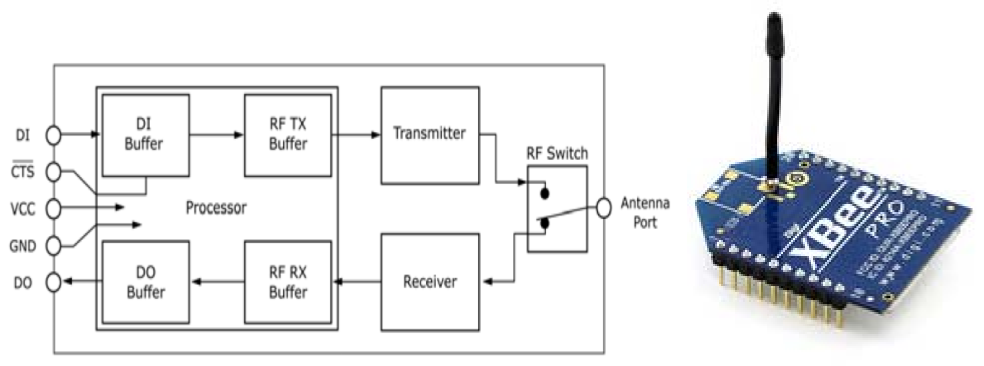


Figure 24 XBee Pro IEEE 802.15.4 Transceiver [After:[19]]

Specifications:

- 2.8 – 3.4 VDC supply voltage
- 215 mA transmit current (max)
- 55 mA idle/receive current
- 10 µA power-down current
- 100 m/1500 m indoor/outdoor range
- 60 mW (18 dBm) transmit power output

- -100 dBm (1% packet error rate) receiver sensitivity
- 250,000 bps raw RF data rate
- 1200-115,200 bps serial interface rate
- 12 DSS channels in the 2.4 GHz band
- 1.5 dBi monopole whip antenna

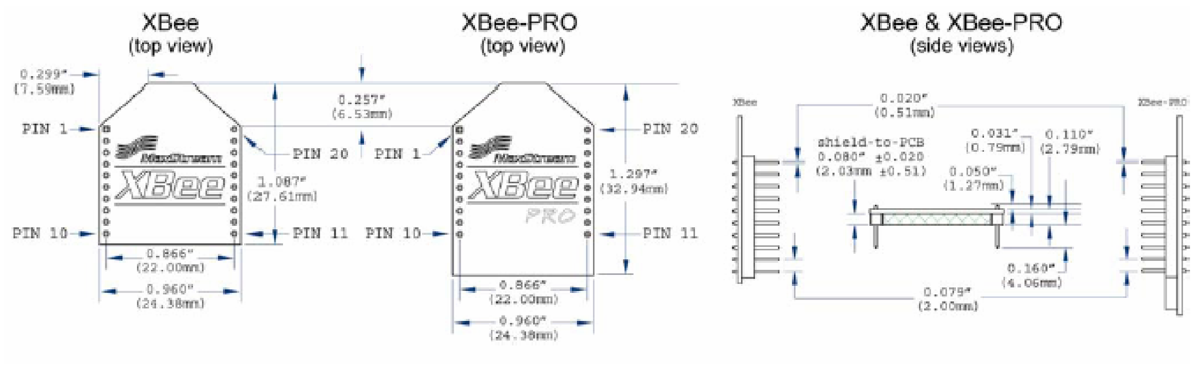


Figure 25 XBee Form Factor [From:[19]]

In the design of the system, the XBee devices serve as the transport medium for the data exchange between the host and the remote camera/servo module. While the XBee module is IEEE 802.15.4 compliant, the device does not incorporate the Beacon-enabled mode discussed in the section describing the IEEE 802.15.4 wireless standard. Instead, this hardware version incorporates the Non-Beacon-enabled mode where associated devices in the network compete individually for use of the medium. In addition, the devices can be configured to operate in a network with or without the use of a network coordinator. A network without a coordinator defines a peer-to-peer topology, while the use of a coordinator defines a point-to-multipoint topology.

The modules offer a high degree of customization through the use of simple “AT” commands familiar in digital communications equipment. This provides a mechanism by which a user can read/set operating parameters within the radio such as destination node address, transmit output power, operating channel, number of ACK failures, and sleep mode characteristics. Configuration parameters can be changed statically, but more importantly, programmatically as the result of a change in the operating environment.

The XBee devices also provide two modes of operation: transparent and application programming interface (API) mode. The transparent mode provides the user with a simple yet effective means of data communication with a remote device. In essence, the transparent mode serves as an abstraction of a serial data line; raw data supplied to the serial port for transmission is packaged into IEEE 802.15.4 frames, sent to the destination device, unpackaged by the remote device, and made available to the user at the other end in the same raw format. Conversely, the API mode of operation provides the user with a wide range of functionality. In this mode, data has to be formatted specifically to match the defined frame types used in the transmission and reception of messages. Figure 26 depicts the structure of one of the frame types, namely a transmit request, used by the XBee transceiver in API mode.

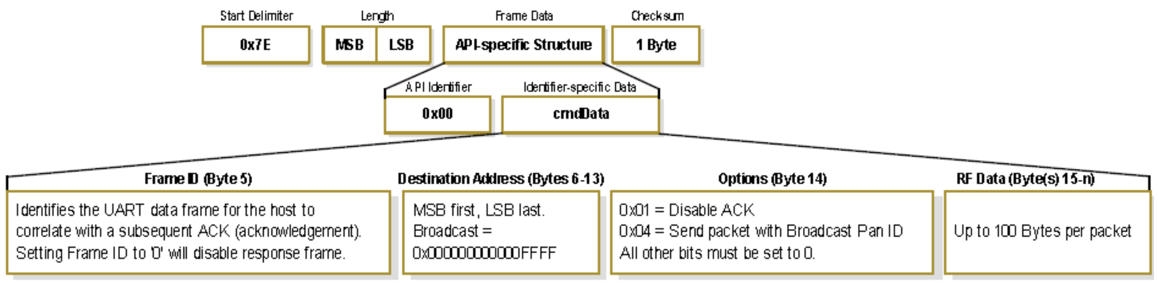

Start Delimiter
0x7E
Length
MSB
LSB
Frame Data
API-specific Structure
Checksum
1 Byte
API Identifier
0x00
Identifier-specific Data
cmdData
Frame ID (Byte 5)
Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). Setting Frame ID to '0' will disable response frame.
Destination Address (Bytes 6-13)
MSB first, LSB last. Broadcast = 0x000000000000FFFF
Options (Byte 14)
0x01 = Disable ACK
0x04 = Send packet with Broadcast Pan ID
All other bits must be set to 0.
RF Data (Byte(s) 15-n)
Up to 100 Bytes per packet

Figure 26 XBee Transmit Request Frame [From:[19]]

While the API mode is more complex, its use provides the user with a wide array of information and increased flexibility. For example, upon the transmission of the frame depicted above, the user is provided with a transmit status frame in return. This frame alone alerts the user to the status of the sent message: successful transmission, no ACK from destination device, or CCA failure. With this information, an application could be programmed to respond accordingly. In addition, receive data frames contain their own unique segments of information beyond the encapsulated data. Information such as the received signal strength (RSSI) and CRC information are carried in each frame and add to the richness of the API operating mode.

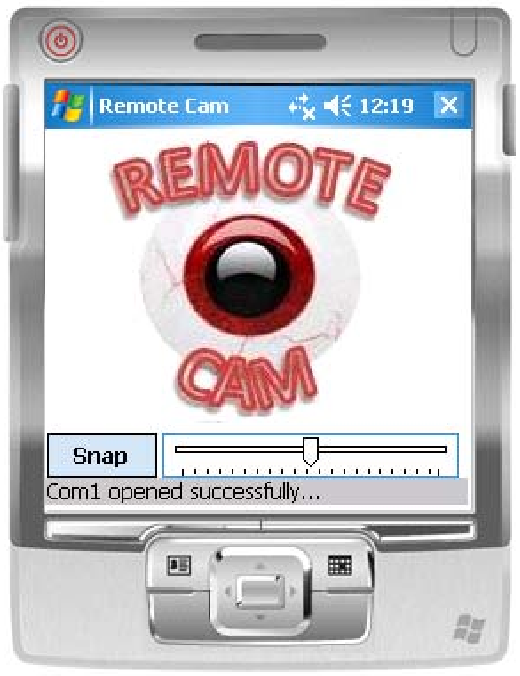
The last point of interest is the XBee's ability to operate in a low-power configuration. While the transmit output power can be modified programmatically, greater power efficiency can be attained by capitalizing on the device's ability to sleep for a maximum period of 268 seconds when associated with a network coordinator. A sleeping device would systematically wake at the end of the defined sleep period and poll the coordinator for any stored messages. If no messages exist, the device reenters the sleep state for another cycle. On the other hand, if the coordinator has information to pass, message transfer operation is initiated and continued until no more data exists to be exchanged. The remote device subsequently returns to the sleep cycle. Since the transceiver consumes less than 10 µA in the sleep state, compared to 55 mA in the idle/receive state, signi ficant power savings can be attained by incorporating such a configuration.

C. SOFTWARE DESCRIPTION AND DISCUSSION

The software component for th e remote camera system consists of two distinct modules. The primary module is the software that controls the graphical user interface (GUI) and the underlying communication protocol on the PDA. This software enables the user to connect with the remote camera via the XBee modules and issue camera, as well as servo position commands. Furthermore, the software is configured to receive the packets of image data, arrange the packets in order, and finally, display the image to the user on the PDA screen. The second software module was developed to run on the servo controller board. This module simply monitors the byte stream for servo-specific commands. Embedded commands are parsed and used in the positioning of the servo motor.

1. PDA Software

The GUI and underlying code for the PDA was developed in Microsoft Visual Studio 2008 in the C# language. The GUI was designed to be simple yet functional in the small amount of screen real estate provided by the HP hw6945. Figure 27 depicts the GUI software, as rendered by the device emulator.



Com1 opened successfully...

Figure 27 Remote Camera GUI Design

This basic functionality includes a 234x162 region to display the image, a track-bar to control the servo position, a single command button to trigger the image capture, a text area to display the system status, and finally a progress bar to show the status of the image download operation. The majority of the underlying code revolved around the passing and parsing of six-byte camera commands that are required for image capture and retrieval. The commands are based on the detailed datasheet provided by the camera manufacturer and are listed in Figure 22 The camera commands are sent and received through the use of a serial port object in the application where it operates at 9600 baud. The most complex operation in the application is the retrieval of the image information stored in the remote camera system. Upon initialization, the camera is instructed to return the image data in packets of a specific size, ranging from 64 to a maximum of 512 bytes. The application was fixed at a packet size of 100 bytes to match the maximum packet size specified by the IEEE 802.15.4 standard. As such, the software was coded to receive each 100-byte packet, append it to the previously received packet, acknowledge receipt to the camera and prepare for the next, if available packet. The cycle is repeated until the image download is complete.

2. Servo Control Software

The code for the servo controller was developed for the OOPic Version C microcontroller using the proprietary IDE and compiler. The first task in the code design for the servo controller was to develop a specific control protocol. Since the control protocol for the camera was well-defined, a simple three-byte protocol was formed that alerted the servo controller that servo position information would follow, {2F, 2F, x} in hexadecimal, where x is the servo position with a value ranging from 1 to 63 decimal that corresponds to the left and right limits of the servo respectively. The servo controller had to listen to the same serial traffic as that of the camera module. That being the case, the servo controller code would have to parse the incoming traffic, and if it did not begin with the servo control specific string, ignore the information and look at the next available string. If and when the controller encountered a valid contr ol string, it simply extracted the servo position byte (value 1 to 63) from the stream and passed it to the servo object for position handling.

The final stage of the hardware and software discussion was the integration of all the components into a functional system that could be used to assess the feasibility of the underlying wireless technology. Custom cables were constructed to gain access to the PDA’s serial port and the interface pins of the XBee device. A simple project case was arranged to house the remote XBee device, servo controller, the servo, and the camera module. The components were connected electrically and battery power was made available as needed. The system was now ready to be tested for range, reliability, and overall functionality.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS

A. SYSTEM DESIGN AND CONGIGURATION

The key to the design, and ultimately the feasibility of this system, is the flexibility provided by the IEEE 802.15.4 enabled XBee transceivers. As stated earlier, the devices provide a large number of configurable parameters that enable the developer to customize the devices to meet very specific operating characteristics. The ability to control the output power, interface data rate, network topology, and the sleep cycle are crucial factors in the design of the system.

- Network Topology: In this design, the network topology that emerged was the point-to-multipoint configuration. This topology resulted from the use of a network coordinator that was ultimately responsible for the management of associated devices. By using the network coordinator configuration, the system was then able to capitalize on the power savings capabilities made available through the sleep cycle functionality.

- Data Transfer Mode: The Xbee transceivers are designed to operate in both transparent (serial line replacement) and API m odes. For this design, it was decided to implement the radios in the transparent mode. This allowed for a less complex programming paradigm of the underlying serial communications process. The API mode would have provided a greater degree of information and control. However, the functionality inherent in the camera module itself, specifically the communication ACKs and NAKs, suffice in the creation of a robust communication scheme.

- Transceiver Output Power: A large factor in the feasibility of the IEEE 802.15.4 standard is the power consumed by a given device. The power output for the XBee transceivers is configurable from a range of 10 dBm up to the highest output of 18 dBm. This system was designed with the output power fixed at the highest level in an attempt to maximize communications range and error free data exchange with the network coordinator. Power savings would come from the sleep cycle scheme.
- Interface Data Rate: While the Xbee transceivers communicate with each other at the fixed, IEEE 802.15.4 defined rate of 250 kbps, connected hosts and/or sensors are able to communicate with the XBees at a rate determined by the developer (1200 up to 115200 baud). In order to accommodate the associated devices (PDA, camera module, servo controller) as well as negate the need for flow control, it was decided to communicate with the XBee modules at the standard rate of 9600 baud, with 8 data bits, no parity, and 1 stop bit. Although relatively slow, the lower data rate prevented buffer

overruns at the XBees and allowed for image retrieval in approximately four seconds based on an average image size of 5 kB.

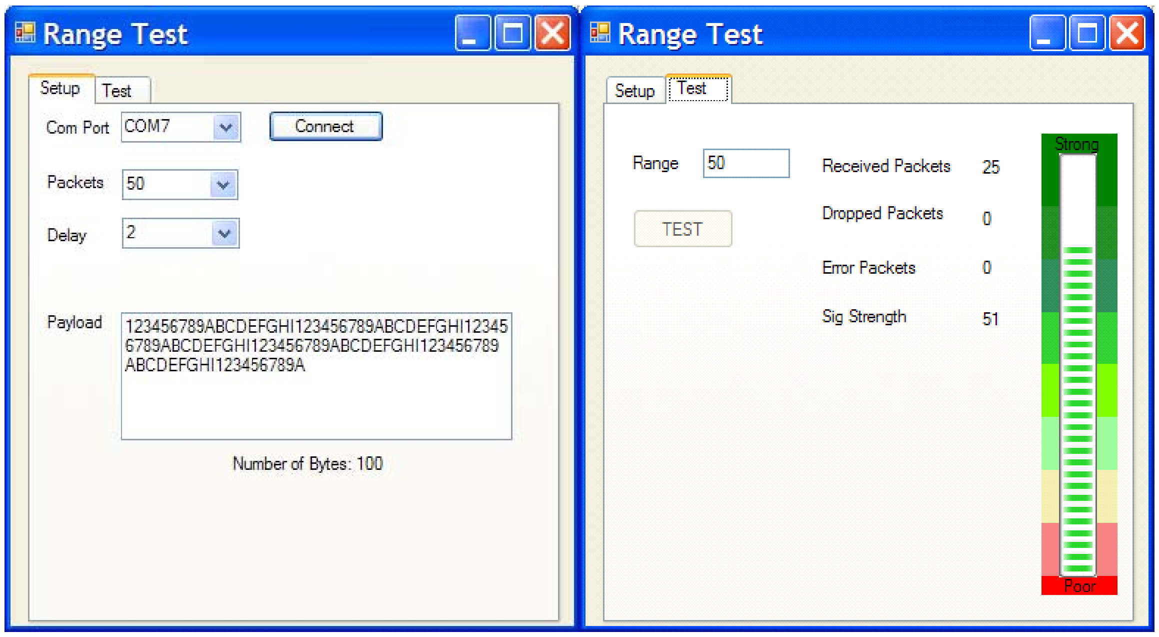
- Sleep Mode Configuration: Establishing a topology with a network coordinator allowed the system to be designed to take advantage of the power saving sleep cycle. The XBee modules can be configured to remain “always on” or sleep from a range of 1 up to 268 seconds. In order to maintain a relatively responsive yet power efficient system, the modules were configured to sleep for a period of four seconds when associated with the coordinator. In this configuration, the module wakes from a sleeping state and polls the coordinator for queued commands. If no messages are pending, the module returns to the sleep state for another four seconds. Conversely, if a command is pending, the module remains awake and completes the data exchange with the coordinator. Upon completion, the module reenters the sleep state. The use of the sleep cycle greatly enhances the low-power characteristics of the XBee modules.

B. RANGE AND RELIABILITY TESTING

In order for the IEEE 802.15.4 standard to be considered a viable solution for its use in wireless sensor networks, the technology must be subjected to a series of tests that can demonstrate its operational capabilities. Most important to an image capture / control system is the range at which the radios can operate and the reliability of the data exchanged between the sensors and the host controller. Consequently, range and bit error rate tests were conducted in an attempt to validate the technology.

1. Range Testing

In order to conduct the range tests, a custom application was constructed that simply transmitted a data packet to the remote device. The remote device would in turn send the received packet back to the originating device. Upon reception, the originating device would inspect the data for errors and extract the signal strength of the received packet. A packet that did not match the sent packet was considered an error packet. If a timeout occurred, the packet was considered a dropped packet. The range and the signal strength of the packet were recorded in a comma separated file. Figure 28 depicts the range test application that was created to conduct and record the results of the testing process while providing the user with feedback on the current operation. The application allowed for a wide range of settings; however, tests were conducted in a manner that closely resembled the operation of the remote camera system; packets were set at 100 bytes each and a total of 10 packets were sent in a given series.



Com Port: COM7
Packets: 50
Delay: 2
Payload: 123456789ABCDEFGHI123456789ABCDEFGHI123456789ABCDEFGHI123456789ABCDEFGHI123456789ABCDEFGHI123456789A
Number of Bytes: 100

Range: 50
Received Packets: 25
Dropped Packets: 0
Error Packets: 0
Sig Strength: 51

Figure 28 Range Testing Application

Range testing was conducted in three, RF-distinct areas: an urban corridor, an urban park, and a remote stretch of beach. The urban corridor was a constricted, yet clear line-of-sight area surrounded by concrete buildings and vegetation. While the length of the corridor provided just over 350 meters, the width was constricted to at most 15 meters. The area was chosen for its high level of RF interference from other 2.4 GHz devices. Also to note was the existence of two large electrical transformers approximately 150 meters down the length of the corridor. The urban park was an area void of buildings but moderately populated by vegetation. The park was surrounded by urban housing and commercial buildings that provided a moderate level of RF interference. The rural stretch of beach was void of buildings, vegetation, and most importantly, RF interference. Figure 29 provides a visual representation of the range test results in terms of the transceiver's signal strength. The range was recorded in 100 meter intervals up to the point where packets were dropped.

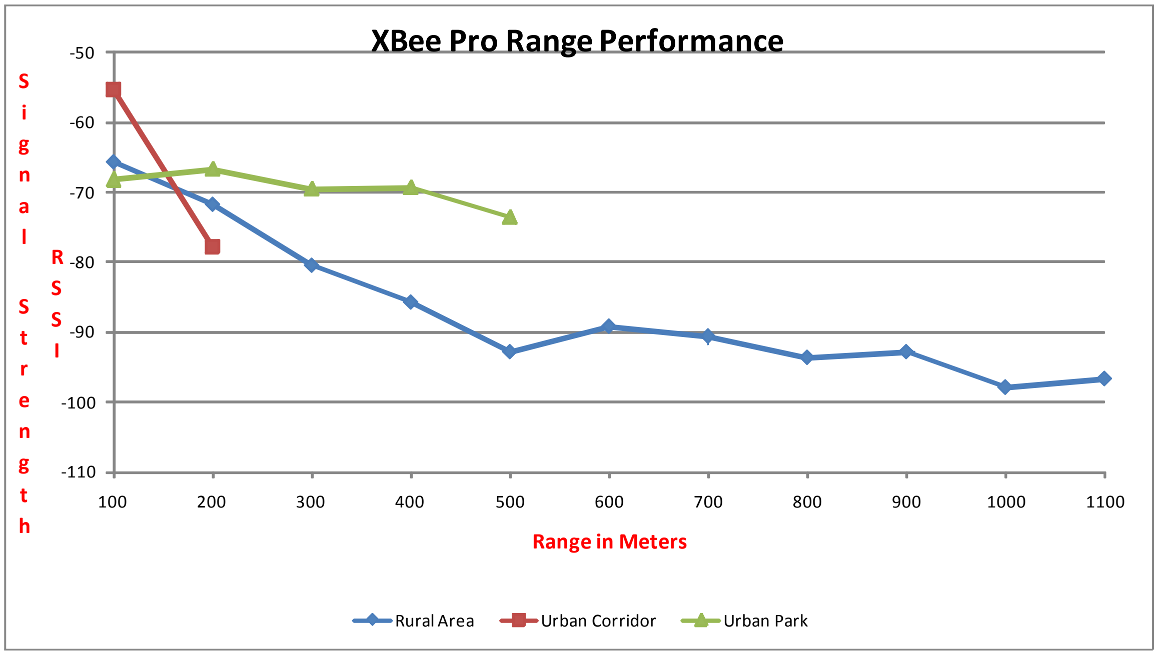
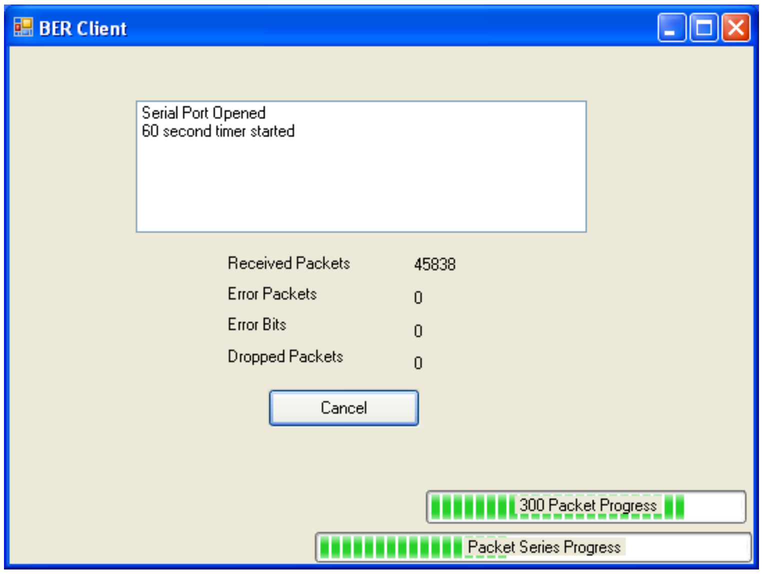


Figure 29   Range Test Results

The XBee devices performed exceptionally well in the rural environment and provided error-free operation up through 1100 meters. Conversely, operation in the urban corridor suffered a significant setback in terms of the maximum range. This outcome followed as expected due to the high degree of competing 2.4 GHz devices as well as the potential interference from the closely located, power transformers. Error-free operation occurred up through 275 meters but quickly deteriorated to a fifty percent drop rate at 300 meters. The final test results from the urban park faired significantly better than the urban corridor. Error-free operation was recorded up through 500 meters.

2. Reliability Testing

In order to conduct the bit error rate (BER) testing, another custom-built application had to be constructed. It is a client / server application where one module runs on a PC (client) and the other runs on a PDA (server). The goal was to create a testing environment where the server, upon request from the client, would send a series of pre-defined data packets (300 packets at 100 bytes per packet with a 100 ms delay between packets) in a given time period to the client. The client, upon receipt of each packet, would inspect the contents of each packet for integrity. A packet that failed the overall integrity check was further scrutinized on a byte-by-byte basis in order to count the individual bit errors that had occurred. This operation was repeated once each minute over the course of six hours in a known hostile environment. Figure 30 shows the operation of the client application near the midpoint of the test run and Figure 31 depicts the potential RF interference caused by 802.11 devices in operation at the time.



```
Serial Port Opened
60 second timer started
```

Received Packets 45838
Error Packets 0
Error Bits 0
Dropped Packets 0

Figure 30 Bit Error Rate Client Application

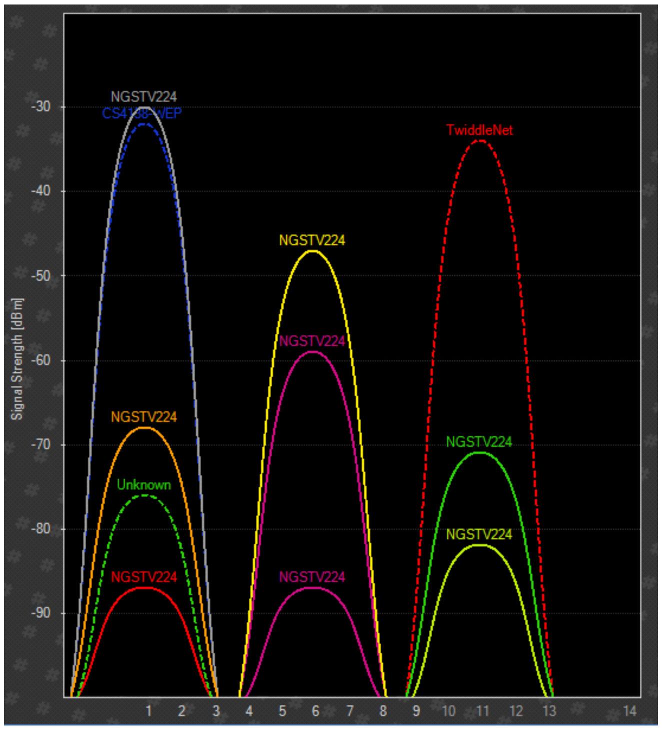


Figure 31 Competing IEEE 802.11 Devices

The datasheet for the XBee device claims a one percent error rate at the bottom end of the receiver's sensitivity rating of -100 dBm. The tests were purposefully conducted in a noisy environment to stress the capabilities of the wireless devices. Figure 31 clearly demonstrates the high levels of RF interference present during the test run. At the conclusion of the six-hour test period, 100 percent of the 108,000 packets sent by the server had been received by the client. More importantly, none of the packets received were received in error; a testament to the reliability of the underlying MAC layer defined by the IEEE 802.15.4 standard.

C. SLEEP CYCLE POWER ANALYSIS

One of the primary attributes of the IEEE 802.15.4 standard is its ability to operate in a manner that allows a device to remain deployed for long periods of time on a simple battery powered configuration. This power saving characteristic was incorporated in the design of the system through the use of the sleep-cycle mechanism described in the system design and configuration section. Although the network coordinator has to r emain "always on" (nonbeacon mode), the associated end devices are able to take advantage of the low-power operation based on the four-second sleep cycle, shown in Figure 32.

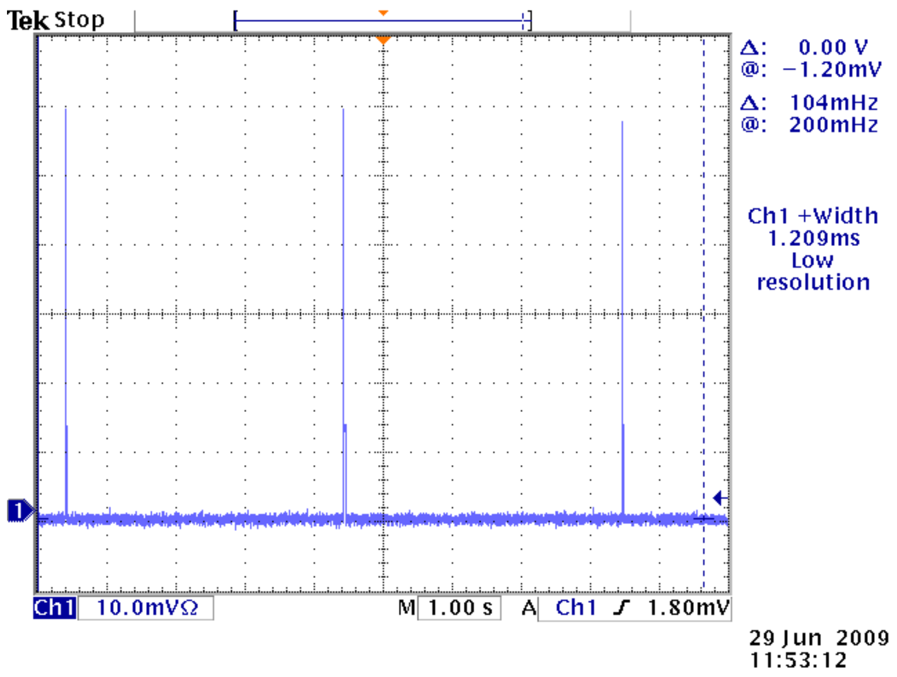


Figure 32 Associated XBee Four Second Sleep Cycle

The diagnostic graphic above represents the current consumption exhibited by a remote device that has been successfully associated with a network coordinator. The current was measured by using the following equipment:

TDS3012B oscilloscope, A6302 current probe, and an AM503A current probe amplifier. The diagram clearly shows the four-second sleep period followed by a short burst of transceiver activity. The power-on portion of the waveform is magnified in Figure 33

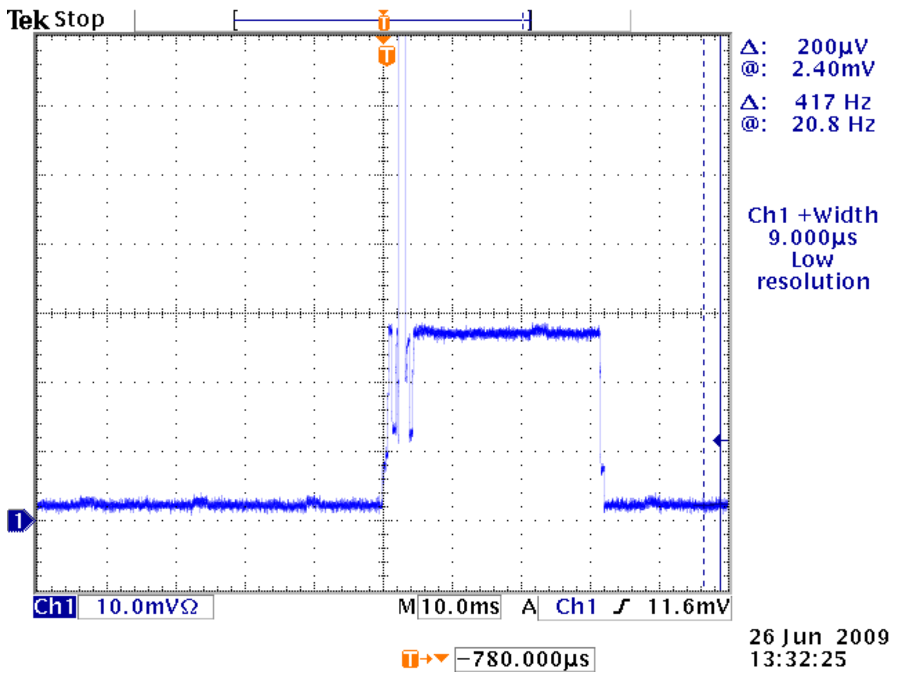


Figure 33 Wake Portion of Four Second Sleep Cycle

This graphic clearly depicts the activity performed by an associated device during the power-on portion of the sleep cycle: switch transceiver to idle mode (55 mA), poll the coordinator for queued data (215 mA), listen for a reply from the coordinator (55 mA), and finally return to the sleep state if no data is pending (10 µA). In the current version of the XBee hardware, the sleep period can be varied from 0 (always on) to 268 seconds. In each setting, excluding always on, the power-on portion of the signal remains the same in amplitude and duration. Therefore, an estimate of the battery life can be calculated to demonstrate the value of the sleep cycle as it relates to device deployment longevity. Table 2 represents several sleep profiles based on the aforementioned signal analysis; a remote device that remains associated with a network coordinator and no data exchange occurs. The baseline calculation represents the profile where a device has been configured to operate continuously in the idle mode. As Table 2 shows, the device could operate for a period of 2.27 days. Conversely, a device configured to sleep for the maximum period of 268 seconds could theoretically operate for 227 months. Finally, the four-second sleep profile implemented in the design of the camera system yields a battery life of more than seven months. These estimates exclude typical battery decay over time.

Profile	Always On (idle)	1 sec	2 sec	4 sec	8 sec	20 sec	50 sec	100 sec	200 sec	268 sec
Sleep Period (in sec)	0	1	2	4	8	20	50	100	200	268
Battery life in hours	54.55	1481.34	2902.86	5705.66	11155.56	26357.89	58176.74	97433.77	147082.35	168927.73
Life in days	2.27	61.72	120.95	237.74	464.81	1098.25	2424.03	4059.74	6128.43	7038.66
Life in months	0.07	1.99	3.90	7.67	14.99	35.43	78.19	130.96	197.69	227.05
*Calculations based on 3000 mAh battery at 3.3 volts, 10 uA sleep current, 55 mA idle current, and 215 mA transmit current										

Table 2 Battery Life Estimate for Associated XBee Module

The tradeoff between a short sleep cycle and a long sleep cycle, excluding current consumption, is the responsiveness of the overarching system. At the maximum sleep period of 268 seconds, an application would have to wait up to 268 seconds, worst case, before any data exchange could occur. Nevertheless, this behavior may be acceptable, even desirable, in a system that has a low-utilization rate, i.e., one that captures a picture once per day.

D. IMAGE RETRIEVAL / SERVO CONTROL

The final stage of testing centered on the performance of the image capture/control system as a whole. The remote camera module was placed at a distance of 300 meters from the host controller in an urban setting with a moderate degree of 2.4 GHz interference. The goal of the test was simply to issue a number of servo position commands followed by an image retrieval command. The cycle was repeated for several series to note the performance of the system. Figure 34 depicts the images captured during the test scenario.

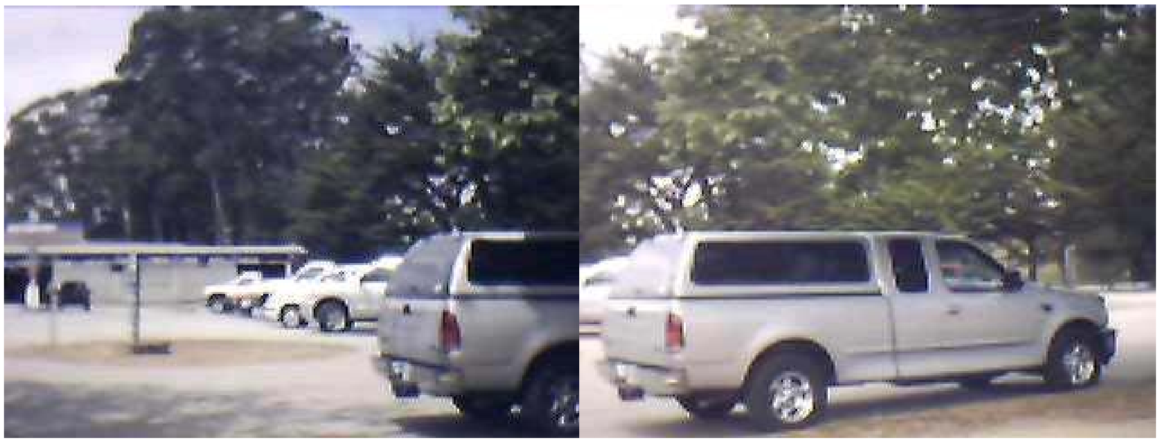


Figure 34 Sample Captured Images

The overall system performed as expected. To note was the potential four-second (maximum) delay that could occur depending on the exact moment the user issued the snapshot or servo position commands. Simply put, the user could have to wait at most four seconds before the system responded; a characteristic of the system design implemented to preserve power. The servo operated in conjunction with the camera without interference. The foreign servo commands were ignored by the camera module and simply discarded without putting the camera in an unstable state. Similarly, the servo controller was not affected by foreign-looking camera commands. The captured images ranged in size from nine to ten KB and varied according to the amount of available light. The image transfer took approximately ten seconds to complete with the default packet size of 100 bytes. Finally, while the quality was marginal at best, the images are quite sufficient to be used in a security application. The underlying IEEE 802.15.4 protocol provided the mechanism for reliable image data and servo control and at the same time did so in a power efficient manner.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND FUTURE WORK

A. SUMMARY

This thesis conducted a feasibility study of the IEEE 802.15.4 wireless standard as the communications medium for a remote image capture / control system. The key characteristics touted in the IEEE 802.15.4 standard include a low-data rate, extremely low-complexity, low-cost, and most importantly, low-power consumption. By integrating the strengths of this technology, along with other off-the-shelf sensors and digital control mechanisms, it was postulated that a robust security application could be constructed to validate the technology as a whole. The hardware that was selected to provide the underlying IEEE 802.15.4 technology was the XBee Pro series transceivers, while the camera system was composed of the C328R camera module, the E-flite digital servo, and the OOPic microcontroller. Accompanying the integration of these components into a functioning system was a series of applications that were designed to facilitate the remaining battery of tests: range testing, bit error rate testing, and power consumption analysis of an associated device.

B. CONCLUSIONS

It was determined through the initial testing that the IEEE 802.15.4 technology is feasible for use in a remote image capture/control application in a wireless sensor network. The simplicity promulgated in the standard allows for the rapid development and deployment of a system that provides reliable data communication. The range, reliability, and the power-saving mechanisms provided by the modules, even in hostile RF environments, makes this platform nearly ideal in this application. Although the range was significantly decreased in an environment plagued by competing RF devices, the ability of the module to automatically switch to a clear or at least less noisy channel, coupled with a multi-hop configuration capability, could easily allow the technology to maintain a long-range data link. Finally, the ability to control the sleep cycle period greatly enhances the low-power characteristic of the standard. A mere four-second sleep cycle allows a device that was previously configured to remain "always on" to extend its battery life from 54 hours to well over seven months.

The use of this technology in a military setting could prove invaluable to the warfighter. Consider the possibilities where such a system was deployed as a means of an early warning or perimeter security application at a remote forward operating base (FOB). The system could be easily expanded to include dozens of remote sensors tethered to a centralized network coordinator by way of the wireless medium. The host controller could be configured to capture updated image data from each sensor in a predetermined manner. More interesting would be the addition of seismic and infrared sensors coupled to the image sensor and IEEE 802.15.4 device, thus providing a means intrusion or suspect behavior detection.   The capability reported in this thesis has great potential to serve the warfighter. Further investigation is warranted. Some potential areas for further study follow.

C. FUTURE WORK

The research conducted in this thesis should be considered an initial investigation into the use of the technology as the backbone for wireless sensor network applications in the realm of image capture / control. As the technology continues to mature, other options become available that can be used to enhance or extend the system. For example, at the time of the research, the XBee devices in use only incorporated the nonbeacon mode in the IEEE 802.15.4 standard. Later stage devices now incorporate the beacon and nonbeacon modes, as well as multi-hop and security related features. Several areas for further investigation are described in the following section.

1. Use of the API Mode to Interface XBee Motes

In the design of the current system, the decision was made to programmatically interface the XBee devices by using the transparent mode of operation. This method provides a simple, high-level mechanism to transfer information to and from the remote transceiver. While simple, this mode does not give the developer access to several key-performance characteristics on the fly, such as the signal strength of the last received packet and more importantly the status of the previously sent packet. The API mode provides data that could then be used by the developer to enhance the performance of the system. For example, instead of relying on a timeout at the application layer, the use of the message status could be used to quickly determine whether or not a remote transceiver received a given packet.

2. Extend Access to the System via the Internet

In the current design of the system, the user controls the camera system by way of the PDA host controller. In an effort to extend the functionality of the system, the host controller could be programmed to act as an Internet gateway to the camera system. A simple client/server application could be developed where the PDA serves as a conduit to the remote camera module by using the PDA’s internal 802.11 and networking capabilities.

3. Investigate Multi-hop Devices

Although the IEEE 802.15.4 standard does not directly specify a multi-hop configuration, mechanisms are in place for the development of such functionality at higher layers (network and application). Later stage XBee devices now incorporate this added functionality in the realm of multi-hop and mesh configurations. The system implemented in this thesis could be expanded to take advantage of this later stage technology and tested to determine its adherence to the low-power / low-data rate premise of the standard and its operability with the C328R image sensor and servo controller.

APPENDIX A. REMOTE CAM SOFTWARE

Appendix A presents the software code for the Remote Cam host application that runs on the PDA.

```
//Remote Cam
//By Dan Krehling
//4/22/09
//Code adapted from Serdar Aktas at:
//http://www.electronics123.com/s.nl/sc.5/category.20/ctype.KB/KB.198867/.f
//The Remote Cam application is a program that runs on a Pocket PC WM5 platform that
//allows the user to connect to, control, and retrieve images from the C328R JPEG camera
//module using the cameras serial communications protocol.  The software also allows the
//user to slave the camera left and right via control signals that are passed to the
//attached servo motor, which operates independently of the camera module.  While the
PDA,
//Camera, and the servo controller can be operated at numerous baud rates, the system
//has been set to operate at 9600 baud.


using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using System.Drawing.Imaging;


namespace CameraPDA
{
    public partial class Form1 : Form
    {   //6 byte global receive array
        Byte[] RcvBuffer = new Byte[6] { 0, 0, 0, 0, 0, 0 };
        //12 byte global receive array
        Byte[] RcvBuffer12 = new Byte[12] { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        //Sync command string
        Byte[] SYNC_ = new Byte[6] { 170, 13, 0, 0, 0, 0 };
        //Expected response after sync command
        Byte[] SYNC_Resp = new Byte[12] { 170, 14, 13, 0, 0, 0, 170, 13, 0, 0, 0, 0 };
        //BaudRate command string(Default 115200 bit/sec)
        Byte[] Baud = new Byte[6] { 170, 7, 15, 1, 0, 0 };
        //Default init command string(Default 320x240)
        Byte[] Init = new Byte[6] { 170, 1, 0, 7, 0, 5 };
        //Default package command string(Default 100 bytes)
        Byte[] Pack = new Byte[6] { 170, 6, 8, 100, 0, 0 };
        //Snapshot command string
        Byte[] Snap = new Byte[6] { 170, 5, 0, 0, 0, 0 };
        //Getpicture command string
        Byte[] GetPic = new Byte[6] { 170, 4, 1, 0, 0, 0 };
        //Expected response after getpicture command
        Byte[] GetPic_Resp = new Byte[12] { 170, 14, 4, 0, 0, 0, 170, 10, 1, 0, 0, 0 };
        //Default ACK command string
        Byte[] ACK = new Byte[6] { 170, 14, 13, 0, 0, 0 };
        //ACK command string to get all the image data
        Byte[] Jpeg_ACK = new Byte[6] { 170, 14, 0, 0, 0, 0 };
        //50 HZ camera light frequency setting
        Byte[] LFreq50hz = new Byte[6] { 170, 19, 0, 0, 0, 0 };
        //60 HZ camera light frequency setting
        Byte[] LFreq60hz = new Byte[6] { 170, 19, 1, 0, 0, 0 };
        bool Response = false;//Command error checker
        bool LastOperation = false;//operation checker for sync, pack, etc
        Int32[] Supported_Bauds = new Int32[2] { 9600, 9600 };//Baudrates to be checked
        Int32 upperbyte = 0, lowerbyte = 0;//Variables to split value into two bytes
        Int16 i = 0;//Baudrate selector & index holder to cut the unneeded bytes
        Int32 counter = 0, counter2 = 0, x = 0, index = 0;//General use counters
        Int32 bytes = 0;//Value that shows the number of bytes of the picture

        //servo control string of "/ / 32" where the "/" signifies servo command
        // and  the int represents the position to move to.
        byte[] ServoControl = new byte[3] { 0x2f, 0x2f, 32 };

        public Form1()
        {
            InitializeComponent();
        }


         private void Form1_Load(object sender, EventArgs e)
        {
            //Center the image when displayed
            pictureBox1.SizeMode = PictureBoxSizeMode.CenterImage;

            //open the comm port
            Connect();
        }

////////////////////////////////////////////////////////////////////////////////////////
/* Function: Connect
 * usage: used to open the serial port on the PDA
* */
        private void Connect()
        {
            //Check if com port 1 is busy
            if (serialPort1.IsOpen)
            {
                //if busy show it to user
                Result.Text = "Com1 is busy. Close other application...";
                Result.Refresh();
                snapButton.Enabled = false;
                progressBar1.Enabled = false;

            }
            else
            {
                // if not open, open port
                serialPort1.Open();
                Result.Text = "Com1 opened successfully...";
                Result.Refresh();
                snapButton.Enabled = true;
                progressBar1.Enabled = true;
            }
        }

///////////////////////////////////////////////////////////////////////////////////////
/* Function: Sync
* usage: used to synchronize the camera module to the host controller at a specific baud
rate
* (hard coded to 9600 baud in this case)
* */
        private void SYNC()
        {
            //Clear global function checker
            Response = false ;

            //Showing action to the user
            Result.Text = "Checking supported baudrate...";
            Result.Refresh();

            //Sending sync command for supported baudrates
            for (i = 0; i != 2; ++i)
            {
                serialPort1.BaudRate = Supported_Bauds[i];

                for (counter = 0; counter != 100 & Response != true; ++counter)
                {
                    //Sending SYNC commands for a few turns and checking if sync acquired
                    Response = SYNC_Command();
                }

                if (Response == true)
                { break; }

            }

            //Show the result to user
            if (Response == true)
            {
                //Order the buttons
                Result.Text = "SYNC...";
                Result.Refresh();
                //this operation was successful
                LastOperation = true;

            }
            else
            {
                //if unsuccesfull show the problem to the user
                Result.Text = "Reset the camera or Retry";
                Result.Refresh();
                LastOperation = false;
            }
        }

////////////////////////////////////////////////////////////////////////////////////////
/* Function: SetJPEGSize
 * usage: used to instruct the camera to take a picture of a specific size, (320X240 in
this case)
 * */
        private void SetJPEGSize()
        {
            //Clear global function checker
            Response = false;

            //Sending init command with the default size of 320X240
            Response = Command(Init);

            //Show the result to user
            if (Response == true)
            {
                //if succesful show user that initialization acquired
                Result.Text = "JPEG size set successfully...";
                Result.Refresh();
                //set global to show last op successful
                LastOperation = true;
            }
            else
            {
                //if unsucessfull show user the problem
                Result.Text = "Unable to set the JPEG size...";
                Result.Refresh();
                LastOperation = false;
            }

        }

////////////////////////////////////////////////////////////////////////////////////////
/* Function: SetFrequency
 * usage: used to instruct the camera use a package size of 512 bytes (other sizes are
available)
```

```
 * */
        private void PackSize()
        {
            //Clear global function checker
            Response = false;

            //Sending package command
            Response = Command(Pack);

            //Show the result to user
            if (Response == true)
            {
                //if succesful show user that initialization acquired
                Result.Text = "Package size set...";
                Result.Refresh();
                //set global to show last op successful
                LastOperation = true;
            }
            else
            {
                //if unsucessfull show user the problem
                Result.Text = "Unable to set the package size...";
                Result.Refresh();
                LastOperation = false;
            }
        }

////////////////////////////////////////////////////////////////////////////////////////
/* Function: SetFrequency
 * usage: used to instruct the camera to operate at the given light frequency (50 or 60
Hertz)
 * */
        private void SetFrequency()
        {
            //Clear global function checker
            Response = false;

            //set the camera frequency to 60hz
            Response = Command(LFreq60hz);

            //Show the result to user
            if (Response == true)
            {
                //if succesful show user that initialization acquired
                Result.Text = "Light freq set...";
                Result.Refresh();
                LastOperation = true;
            }
            else
            {
                //if unsucessfull show user the problem
                Result.Text = "Unable to set the light frequency...";
                Result.Refresh();
                LastOperation = false;
            }
        }

////////////////////////////////////////////////////////////////////////////////////////
/* Function: SnapShot
 * usage: used to instruct the camera module to take and store a photo in the onboard
memory
 * */
        private void SnapShot()
        {
            //Clear global function checker
            Response = false;

            //Sending snapshot command
            Response = Command(Snap);
            //Show the result to user
            if (Response == true)
            {
                //if succesful show user that initialization acquired
                Result.Text = "Snapshot taken...";
                Result.Refresh();
                LastOperation = true;
            }
            else
            {
                //if unsucessfull show user the problem
                Result.Text = "Unable to take the picture...";
                Result.Refresh();
                LastOperation = false;
            }
        }



////////////////////////////////////////////////////////////////////////////////////////
/* Function: Preview
 * usage: used to instruct the camera module to start transferring the image data to the
 * host.
 * */
        private void Preview()
        {
            Int32 Response = 0;

            Response = Preview_Command();

            if (Response != 0)
            {   //if succesful show user that initialization acquired
                Result.Text = "Image Size: " + Response.ToString() + " bytes";
                Result.Refresh();
                Construct_JPEG(Response);

            }
            else
            {   //if unsucessfull show user the problem
                Result.Text = "Unable to preview the picture...";
                Result.Refresh();

            }
            //reset and hide the progress bar
            progressBar1.Value = 0;
            progressBar1.Visible = false;

        }



////////////////////////////////////////////////////////////////////////////////////////
 /* Function: SYNC_Command
 * usage: used to acquire synchronization between the c328 camera.Must be used first
 * */
        bool SYNC_Command()
        {
            //Arranging acknowledge
            ACK[2] = SYNC_[1];

            //Sending sync command
            serialPort1.Write(SYNC_, 0, 6);
            Thread.Sleep(50);

            //Checking if receive buffer is empty?
            if (serialPort1.BytesToRead != 0)
            {
                Thread.Sleep(1);
                serialPort1.Read(RcvBuffer12, 0, 12);
                //checking if true response received
                if (RcvBuffer12[0]==SYNC_Resp[0] & RcvBuffer12[1]==SYNC_Resp[1] &
                    RcvBuffer12[2]==SYNC_Resp[2] & RcvBuffer12[4]==SYNC_Resp[4] &
                    RcvBuffer12[5]==SYNC_Resp[5] & RcvBuffer12[6]==SYNC_Resp[6] &
                    RcvBuffer12[7]==SYNC_Resp[7] & RcvBuffer12[8]==SYNC_Resp[8] &
                    RcvBuffer12[9]==SYNC_Resp[9] & RcvBuffer12[10]==SYNC_Resp[10] &
                    RcvBuffer12[11]==SYNC_Resp[11] )
                {
                    // if yes sending ack command and returning true
                    serialPort1.Write(ACK, 0, 6);
                    return true;
                }
                else
                {
                    //else sending false
                    return false;
                }
            }

            return false;

        }



////////////////////////////////////////////////////////////////////////////////////////
/*Function name :Preview_Command
* usage: used to preview the snapshot picture
* */
       Int32  Preview_Command()
       {

           //Writing getpicture string to com port
           serialPort1.Write(GetPic, 0, 6);
           Thread.Sleep(100);

           //Checking receive buffer
           if (serialPort1.BytesToRead != 0)
           {
               //Reading the receive buffer
               serialPort1.Read(RcvBuffer12, 0, 12);
           }


           //Checking if true response received
           if ( RcvBuffer12[0] == GetPic_Resp [0] & RcvBuffer12[1] == GetPic_Resp [1] &
                RcvBuffer12[2] == GetPic_Resp [2] & RcvBuffer12[4] == GetPic_Resp [4] &
                RcvBuffer12[5] == GetPic_Resp [5] & RcvBuffer12[6] == GetPic_Resp [6] &
                RcvBuffer12[7] == GetPic_Resp [7] & RcvBuffer12[8] == GetPic_Resp [8] )
           {
               //if yes sending the image data length
               bytes = Convert.ToInt32(RcvBuffer12[11]);
               bytes <<= 8;
               bytes = bytes | Convert.ToInt32(RcvBuffer12[10]);
               bytes <<= 8;
               bytes = bytes | Convert.ToInt32(RcvBuffer12[9]);
               return bytes;
           }
           else
           {
               //else sending zero
               return 0;
           }
       }

////////////////////////////////////////////////////////////////////////////////////////
/* Function:Construct_JPEG(Response);
* usage: used to build the image from the data returned from the camera
* */
       void Construct_JPEG(Int32 bytes)
```

```
        {

            Byte[] Jpeg = new Byte[bytes];   //Null jpeg stream

            //set the upper bound of the progress bar to the value "bytes" passed in above
            progressBar1.Maximum = bytes;
            progressBar1.Value = 0;
            progressBar1.Visible = true;

            //Clearing the global counters
            counter = 0; counter2 = 0; x = 0; index = 0 ;

            //Clearing the upper and lower bytes
            upperbyte = 0; lowerbyte = 0;


            for ( ;counter != bytes ;  )   //loop until getting all image data
            {

                Jpeg_ACK[4] = Convert.ToByte(lowerbyte);
                Jpeg_ACK[3] = Convert.ToByte(upperbyte);

                // send the acknowledge command
                serialPort1.Write(Jpeg_ACK, 0, 6);
                //pause while the camera module sends image data to the buffer
                Thread.Sleep(400);
                if ( (x = serialPort1.BytesToRead) != 0)
                {
                    //read the values
                    Byte[] Temp = new Byte[serialPort1.BytesToRead];
                    serialPort1.Read(Temp, 0, serialPort1.BytesToRead);

                    //control if true package got
                    if (Temp[0] == lowerbyte & Temp[1] == upperbyte)
                    {
                        for (int i = 4 ; i != x - 2; ++i)
                        {
                            //cut the unused bytes
                            Jpeg[index] = Temp[ i ];
                            ++index;
                        }

                        //set the progress bar
                        progressBar1.Value = index;
                    }
                    else
                    {
                        // if error occurs show it to user
                        Result.Text = "Corruption occurred..";
                        Result.Refresh();
                        break;
                    }

                    //arrange the controllers
                    counter = counter + x - 6;
                    ++counter2;
                    upperbyte = counter2 >> 8;
                    lowerbyte = counter2 & 0x00FF;
                }


            }

            // Send the last command
            Jpeg_ACK[4] = Convert.ToByte(lowerbyte);
            Jpeg_ACK[3] = Convert.ToByte(upperbyte);
            serialPort1.Write(Jpeg_ACK, 0, 6);

            string fileName = "\\Temp\\PIC" + DateTime.Now.TimeOfDay.Seconds.ToString() +
".jpg";
```

```
            //Display the picture
            Stream image = new MemoryStream(Jpeg);
            Stream fs = new FileStream(fileName, FileMode.CreateNew);

            pictureBox1.Image = new Bitmap(image);
            //save the image to the Temp\PIC directory
            pictureBox1.Image.Save(fs,ImageFormat.Jpeg);
        }


//////////////////////////////////////////////////////////////////////////////////////////
/* Function: Command
 * usage: used to pass specific commands to the camera module
 * */
        public bool Command(Byte[] Parameter)
        {

            //Arranging acknowledge
            ACK[2] = Parameter[1];

            //Sending parameters to com1
            serialPort1.Write(Parameter, 0, 6);

            //Waiting the device to send response
            Thread.Sleep(400);

            //Checking receive buffer
            if (serialPort1.BytesToRead != 0)
            {
                Thread.Sleep(1);

                //Reading the receive buffer
                serialPort1.Read(RcvBuffer, 0, 6);
            }


            //Show the result to user
            if (RcvBuffer[0] == ACK[0] & RcvBuffer[1] == ACK[1] & RcvBuffer[2] == ACK[2] &
                RcvBuffer[4] == ACK[4] & RcvBuffer[5] == ACK[5])
            {
                return true;
            }
            else
            {
                return false;
            }

        }

//////////////////////////////////////////////////////////////////////////////////////////
/* Function: snapButton_Click
 * usage: event that fires when the user clicks the SNAP button on the interface.  From
this
 * command, most of the other camera functions are called into action with the result
being
 * the image rendered on the screen.
 * */
        private void snapButton_Click(object sender, EventArgs e)
        {
            snapButton.Enabled = false;
            SYNC();
            if (LastOperation) SetJPEGSize();
            if (LastOperation) PackSize();
            if (LastOperation) SetFrequency();
            if (LastOperation) SnapShot();
            if (LastOperation) Preview();
            snapButton.Enabled = true;

        }
```

```
////////////////////////////////////////////////////////////////////////////////////////
/* Function: Form1_Closing
 * usage: closes out the serial port connection when the form is closed
 * */
       private void Form1_Closing(object sender, CancelEventArgs e)
       {
           if (serialPort1.IsOpen)
           {
               serialPort1.Close();
           }
       }

////////////////////////////////////////////////////////////////////////////////////////

/* Function: SetFrequency
 * usage: used to send position information to the servo when the user commands via the
 * track bar on the interface.
 * */
       private void trackBar1_ValueChanged(object sender, EventArgs e)
       {
           ServoControl[2] = (byte)(64 - trackBar1.Value);
           serialPort1.Write(ServoControl, 0, 3);
           Result.Text = "Servo Pos: " + trackBar1.Value.ToString();
           Thread.Sleep(400);
           //clear anything the camera may have returned as a result of this last command
           serialPort1.DiscardInBuffer();
       }

    }//end class Form1

       }//end namespace CameraPDA
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. SERVO CONTROL SOFTWARE

Appendix B presents the software code for the OOPic operation of the remote servo.

```
'This program listens to the serial port traffic
'for a special code sequence "2F,2F, X" hexadecimal
'where the X is a value ranging from 1 to 63.  When
'encountered, the program captures the X value and
'positions the servo object to that position.

Dim A As New oSerialPort
Dim B As New oByte
Dim S1 As New oServo


Sub Main()
  A.Baud = cv9600
  A.Operate = cvTrue
  S1.IOLine = 24
  S1.Center = 28
  S1.Value = 32
  S1.Operate = cvTrue


  Do
    If A.Received = cvTrue Then
       B.Value = A.Value
       'look for the first occurrence of "2F" hex
       If B.Value = 0x2f Then
               If A.Received = cvTrue Then
                     B.Value = A.Value
                     'look for the second occurrence of "2F" hex
                     If B.Value = 0x2f Then
                        If A.Received = cvTrue Then
                           B.Value = A.Value
                           'position the servo to the X value
                           S1.Value = B.Value
                        End If
                      End If
               End If
             End If
    End If
  Loop
End Sub
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] W. Roush, A. M. Goho, E. Scigliano, D. Talbot, M. M. Waldrop, G. T. Huang, P. Fairley, E. Jonietz and H. Brody, "10 Emerging Technologies," ***Technol. Rev.,*** vol. 106, pp. 33, 02. 2003.

[2] E. H. Callaway, ***Wireless Sensor Networks: Architectures and Protocols.*** Boca Raton, Fl: CRC Press, 2004,

[3] D. Estrin, D. Culler, K. Pister and G. Sukhatme, "Connecting the Physical World with Pervasive Networks," ***IEEE Pervasive Computing,*** vol. 1, pp. 59-69, 2002.

[4] T. Haenselmann. "Sensornetworks," ***An FDL'd Textbook on Sensor Networks,*** [Online]. Available: http://www.informatik.uni-mannheim.de/~haensel/sn_book/. [Accessed: Mar. 14, 2009].

[5] K. Romer and F. Mattern, "The Design Space of Wireless Sensor Networks," ***IEEE Wireless Communications,*** vol. 11, pp. 54-61, 2004.

[6] J. Pike. "Sound Surveillance System (SOSUS)," ***globalsecurity.org,*** [Online]. Available: http://www.globalsecurity.org/intell/systems/sosus.htm. [Accessed: Jan. 25, 2009].

[7] C. Y. Chong, S. P. Kumar and B. A. Hamilton, "Sensor Networks: Evolution, Opportunities, and Challenges," ***Proc IEEE,*** vol. 91, pp. 1247-1256, 2003.

[8] K. Martinez, P. Padhy, A. Elsaify, G. Zou, A. Riddoch, J. K. Hart and H. L. R. Ong, "Deploying a Sensor Network in an Extreme Environment," ***IEEE SUTC,*** vol. 1, pp. 186-193, 2006.

[9] R. Beckwith, D. Teibel, P. Bowen and I. Res, "Unwired wine: Sensor networks in vineyards," in ***Sensors, 2004. Proceedings of IEEE,*** 2004, pp. 561-564.

[10] Anonymous "Pervasive 2004, April 18-23, Linz / Vienna, Austria," Available: http://www.pervasive2004.org/program hotspotpapers.php. [Accessed: May 1, 2009].

[11] J. A. Gutierrez, E. H. Callaway and R. Barrett, ***Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors with IEEE 802.15.4.*** New York, NY: IEEE Press, 2004.

[12] The Institute of Electrical and Electronics Engineers, Inc, "IEEE Standard for Information technology- Telecommunications and information exchange between systems- Local and metropolitan area networks- Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)," ***IEEE Std 802. 15. 4-2006 (Revision of IEEE Std 802. 15. 4-2003),*** pp. 1-305, 2006.

[13] E. Callaway, P. Gorday, L. Hester, J. A. Gutierrez, M. Naeve, B. Heile and V. Bahl, "Home Networking with IEEE 802.15.4: A Developing Standard for Low-rate Wireless Personal Area Networks," ***IEEE Communications Magazine,*** vol. 40, pp. 70-77, 2002.

[14] P. Kumar, M. Günes, A. A. B. Al Mamou and J. Schiller. "Real-time, Bandwidth, and Energy Efficient IEEE 802.15. 4 for Medical Applications," [Online]. Available: http://cst.mi.fu-berlin.de/publications/pdf/2008-kumar-FGSN.pdf. [Accessed: Apr. 12, 2009].

[15] Inc Hewlett Packard. "HP iPAQ hw6900 Mobile Messenger Series," ***HP.com,*** [Online]. Available: http://h20000.www2.hp.com/bizsupport/TechSupport/Document.jsp?objectID=c00699792&lang=en&cc=us&taskId=101&prodSeriesId=1822489&prodTypeId=215348. [Accessed: June 15, 2009].

[16] Inc Savage Innovations. "OOPic Home Page," ***oopic.com,*** [Online]. Available: http://www.oopic.com/con40.htm#Description. [Accessed: June 22, 2009].

[17] Inc Horizon Hobby. "E-flite - Advancing Electric Flight," ***e-fliterc.com,*** [Online]. Available: http://www.e-fliterc.com/Products/TechnicalSpecs.aspx?ProdID=EFLRDS75H. [Accessed: June 23, 2009].

[18] COMedia Ltd. "COMedia Website," ***comedia.com,*** [Online]. Available: http://www.comedia.com.hk/fp2008/Spec_PDF/C328R_UM.pdf. [Accessed: June 17, 2009].

[19] Inc MaxStream. "XBee® & XBee-PRO® 802.15.4 OEM RF Modules Datasheet," ***User's Manual,*** [Online]. Available: http://ftp1.digi.com/support/documentation/90000982_A.pdf. [Accessed: June 19, 2009].

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California

4. Director, Training and Education, MCCDC, Code C46
   Quantico, Virginia

5. Director, Marine Corps Research Center, MCCDC, Code C40RC
   Quantico, Virginia

6. Marine Corps Tactical Systems Support Activity
   (Attn: Operations Officer)
   Camp Pendleton, California